Decodificador de texto Antiguo:

Cicerón

Guillermo de la Horra Roda
Madrid 2011

UNIVERSIDAD CARLOS III
CAMPUS DE LEGANÉS, MADRID
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

Decodificador de texto Antiguo: Cicerón

Memoria de Proyecto fin de Carrera de Ingeniería Técnica Industrial en la especialidad de Electrónica Industrial
Presentada por Guillermo de la Horra Roda

Directores del proyecto:

Prof. Dr. Luis Moreno Lorente
Prof. Dra. Pilar Azcarate Aguilar-Amat

Madrid, mayo 2011

1. Agradecimientos

Agradezco al Dr. Luis Moreno Lorente el haberme brindado la posibilidad de realizar este proyecto, cuya ejecución me ha ayudado a aprender y profundizar en el campo de la identificación y procesamiento de imágenes. Gracias a sus acertados consejos, se ha podido concluir este trabajo.

Así mismo agradezco a la Dra. Pilar Azcarate Aguilar-Amat su gentileza al aceptar la codirección de este proyecto, dotándolo de un carácter humanista.

Al profesor Don Fernando García Fernández, sus apreciadas enseñanzas y sus valiosas orientaciones.

Una palabra debe ser vestida como una diosa y elevarse como un pájaro

Proverbio tibetano

Índice *Página*

2. Objetivos y Motivación del proyecto:

El objetivo de este proyecto es desarrollar un programa informático que permita reconocer, de forma eficiente y fiable, grafías antiguas. En concreto, se ha trabajado con alguna de las modalidades medievales de escritura contenidas en el códice nº 91 de la Sección de Códices del Archivo Histórico Nacional (AHN). Además, el programa ha de ser capaz de transcribir dichas grafías a un alfabeto legible, que facilite su estudio y análisis. El proyecto se ha estructurado en dos grandes bloques. En el primero se trata de contextualizar el objeto de la investigación, mediante una exposición general sobre la evolución histórica de los soportes de escritura, de las formas y características de los libros y de las escrituras medievales. Para ello, nos hemos basado en las obras que aparecen citadas en la bibliografía. En el segundo bloque, se abordará propiamente el objetivo de este proyecto.

La razón de haberlo llevado a cabo ha sido la inexistencia actualmente de un programa que realice una tarea similar, es decir, el reconocimiento de las grafías de un códice del medievo y su posterior transcripción.

Para la realización del proyecto se ha decidido utilizar Microsoft Visual Studio 2008 Professional Edition. El motivo ha sido la mayor familiaridad y compatibilidad que suponía el sistema operativo Windows. El lenguaje utilizado ha sido C. Además, debido a la naturaleza del proyecto, se han utilizado las librerías multiplataforma OpenCv de Intel, ya que cuentan con numerosas funciones para la trata de imágenes. Dichas funciones han simplificado en parte la tarea iterativa de los algoritmos empleados. Las librerías OpenCv están publicadas bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas. El soporte informático empleado ha sido un ordenador Acer Aspire X3900.

3. Introducción

3.1 AHN, Diversos y Colecciones. Códices, nº 91

Este manuscrito procede del monasterio cisterciense de Santa María de Rioseco, ubicado en el Valle de Manzanedo, actual provincia de Burgos, junto al río Ebro, en el territorio donde tuvo su origen el primitivo Condado de Castilla.

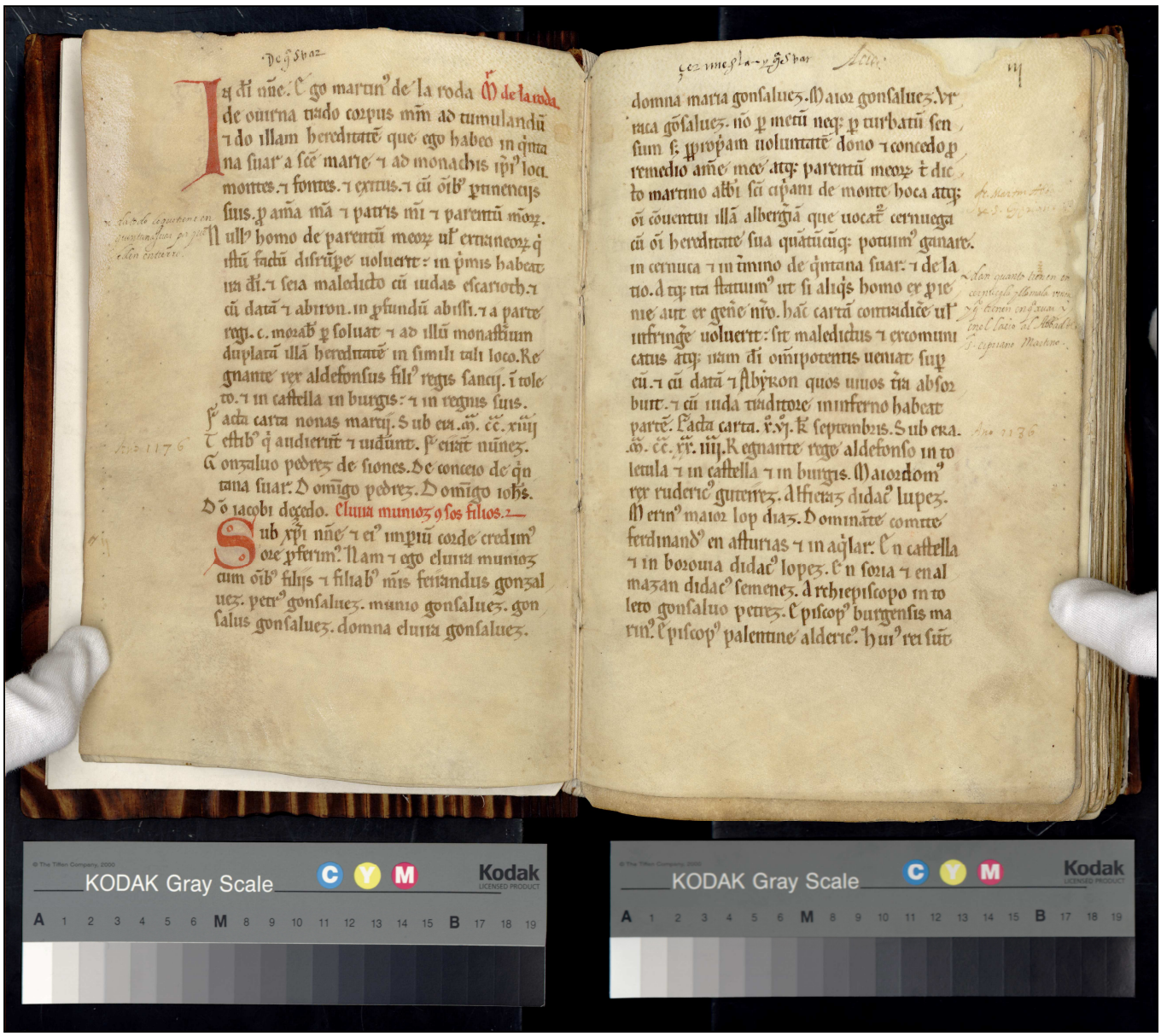
En el códice se copiaron 180 documentos del archivo monástico. La mayoría corresponden a los siglos XII y XIII, con escasos añadidos de los siglos XIV y XV y algunos documentos sin datar.

Tanto las imágenes como las transcripciones que se reproducen a continuación han sido cedidas por CRELOC[1].

[1] *Clientela y redes locales en la Castilla medieval (ss. XI-XIV). Estudio histórico y tecnologías documentales* CRELOC (I y II). Proyectos financiados por el Ministerio de Ciencia y Tecnología (Referencias BHA2003-03039 y HUM2006-04544/HIST y dirigidos por la Dra. Cristina Jular Pérez-Alfaro, a la que agradecemos la cesión de dichos materiales.

3.1.1 Muestra de Imágenes del Manuscrito

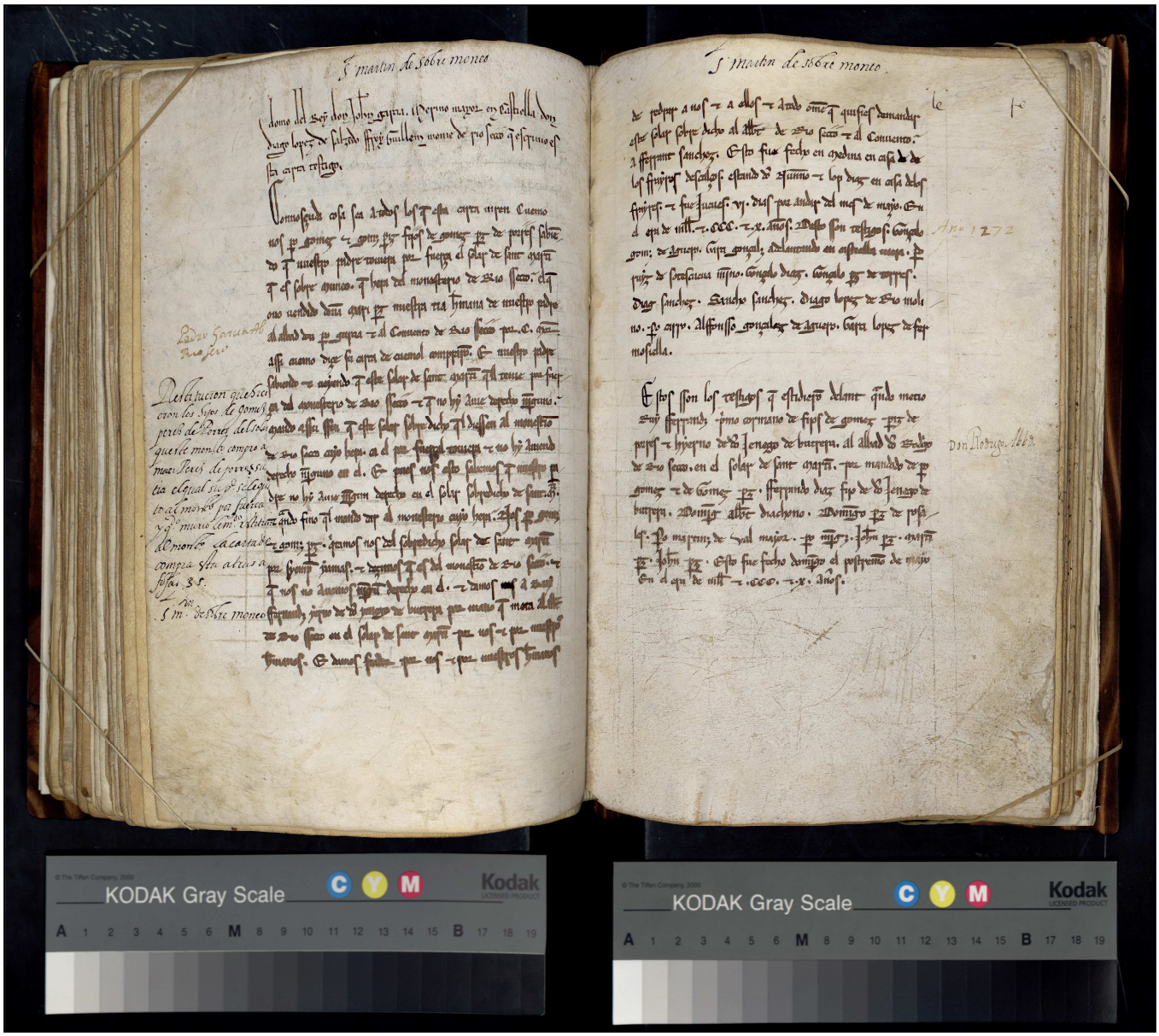
A02v03r



A16v17r

A08v09r

A68v69r

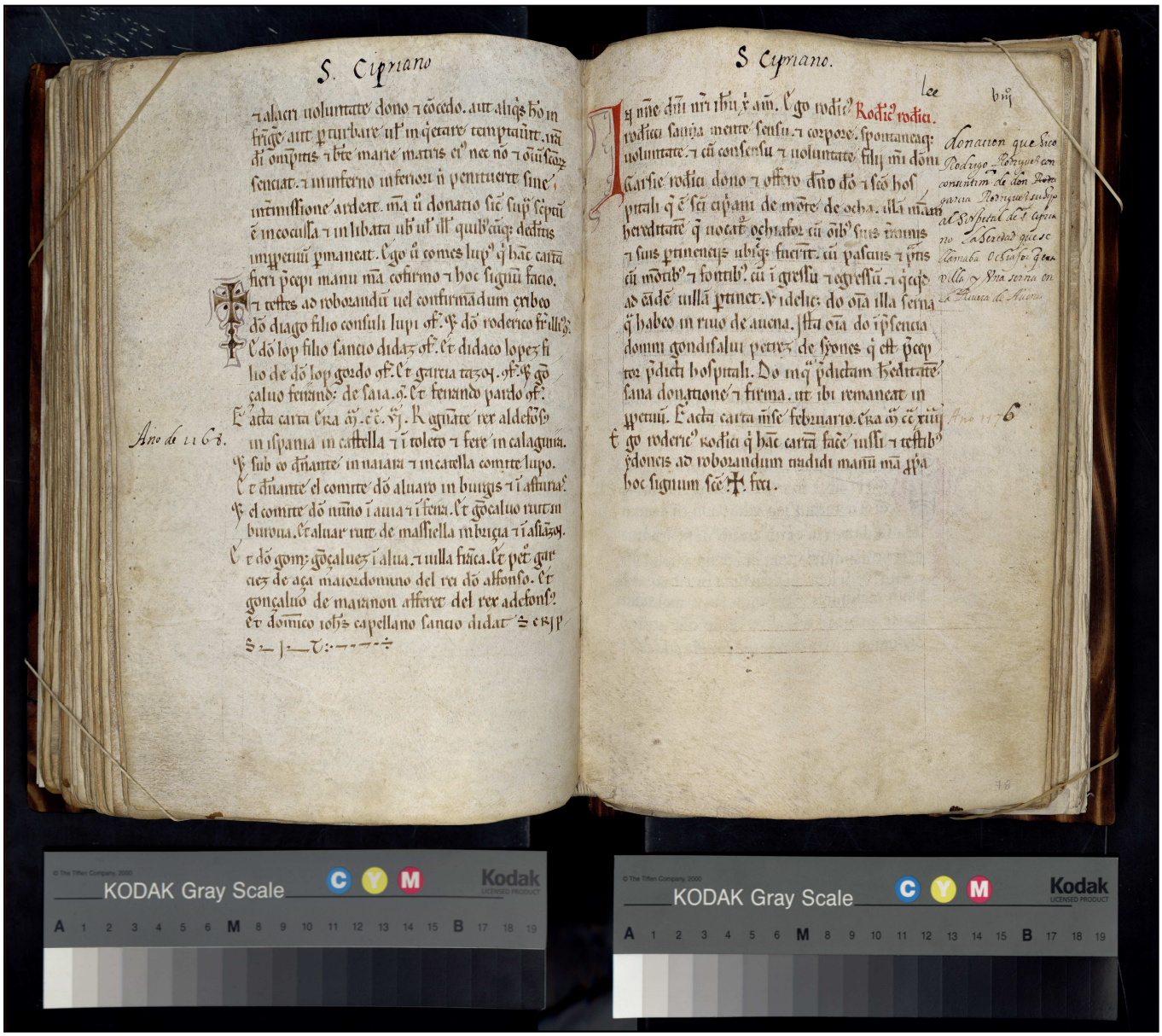


A69v79r

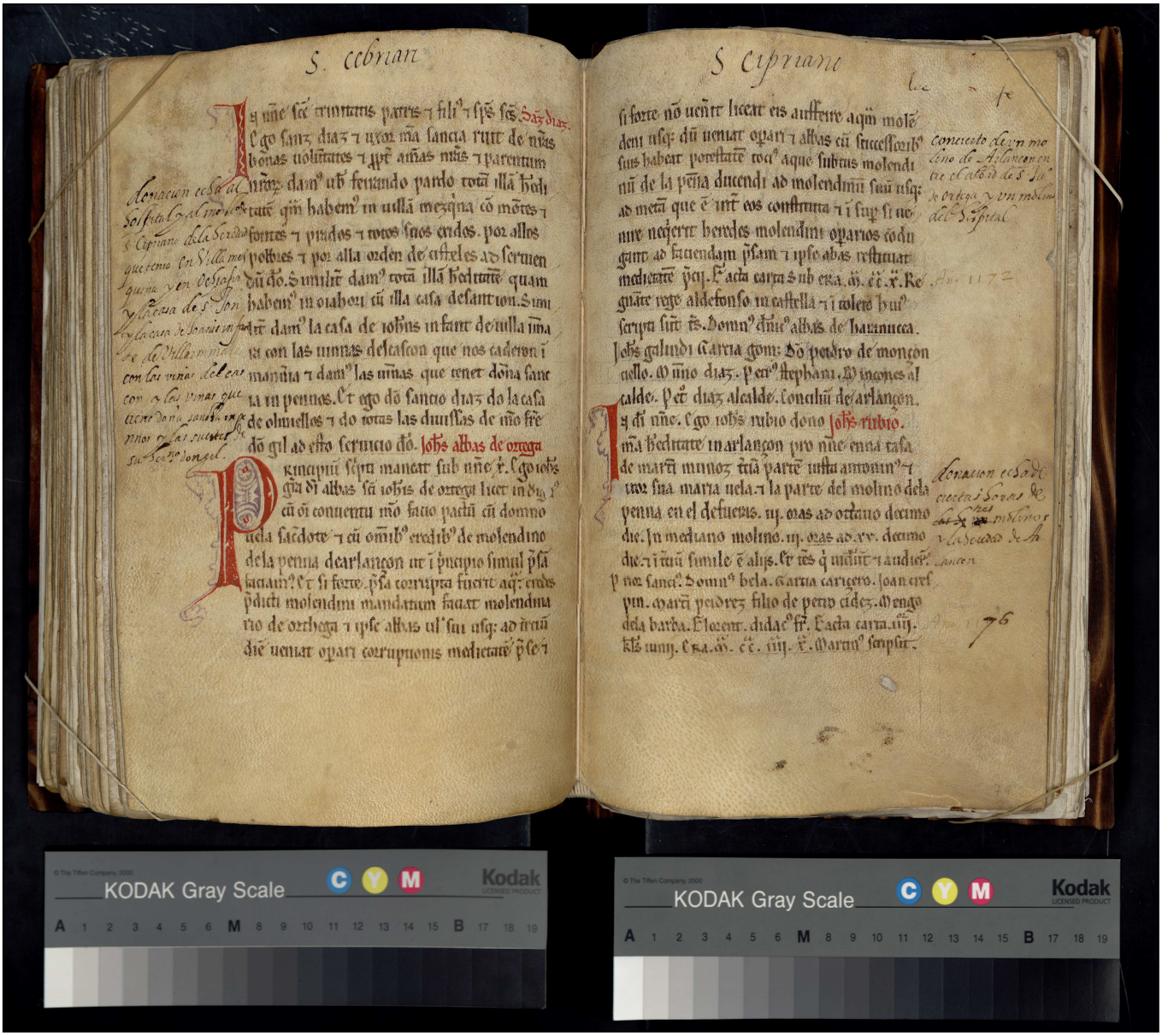
A70v71r

A72v73r

A77v78r


S. Cipriano
S Cipriano.
Rodrigo rodiz.
Ano de 1168.
Ano 1176
KODAK Gray Scale
Kodak

A78v79r



A90vT01r


Nº R-831
1
Tabla delos Privilegios cartas de Venta, trueques, cambios y posesiones
eneste libro.
A
Andeno
Ay Un trueque y cambio de Un Suelo en
Toledo de la Corona en Andeno al
Monast.º
Sedo
Ay Una carta de Venta delas Heredades
Solares y Suelo en Sedo delas
dichas
Sedo
Ay Una Carta de Venta delas tierras y Pueblo
Malvec en Sedo delas herederas
carta de Venta esta abras fº 36
Abasas.
Ay Una carta de Venta dela Ygleª de
Sedo manera tenia en Ygleª abasas
Abasas.
Ay Un Padron delas Heredades y Pecheros
tiene en dhas Villas de Abasas.
Alda
Ay Una carta de Como de Onsalvo
de la Casa Santa de S. millan dondesta
ha Un mesmo concordia que Una Vasalla.
Alda.
Ay Una donacion de todo la Heredad
que tenia en Alda con condicion que
se sara alli la Alda
Ramos los Vasallos y damos todos
q tiene sin Reservo ni Pecho Real
Salvo
Ay Un memorial delas heredades y Pue
Monast.º tiene en andeno
A. H. N.
CODICES
KODAK Gray Scale
Kodak

3.1.2  ranscripción de los documentos

A continuación se ofrecen las transcripciones de algunos de los documentos del manuscrito presentes en el epígrafe anterior.

3.1.2.1 Imagen A02v03r

fol. 2v.

1176, marzo, 7.

(*En tinta roja:)* Martin de la Roda.

In Dei nomine. Ego Martinus de la Roda de Ouirna trado corpus meum ad tumulandum et do illam hereditatem que ego habeo in Quintana Suar a Sancte Marie et ad monachis ipsius loci, montes et fontes et exitus et cum omnibus pertinenciis suis, pro anima mea et patris meis et parentum meorum. Nullus homo de parentum meorum uel extraneorum qui istud factum disrumpere uoluerit, in primis habeat ira Dei et seia maledicto cum Iudas Escarioth et cum Datan et Abiron in profundum abissi et a parte regi C morabetinos persoluat, et ad illum monasterium duplatam illam hereditatem, in simili tali loco.

Regnante rex Aldefonsus filius regis Sancii in Toleto et in Castella, in Burgis et in regnis suis. Facta carta nonas marcii sub era Mª CCª XIIIIª.

Testibus qui audierunt et uiderunt: Ferrant Nunnez, Gonzaluo Pedrez de Siones. De conceio de Quintana Suar: Domingo Pedrez, Domingo Iohannis, Don Iacob Deçedo.

fols. 2v-3v.

1186, agosto, 17.

(*En tinta roja:*) Eluira Munioz con sos filios.

Sub Christi nomine et eius imperium corde credimus ore. Nam et ego Eluira Munioz cum omnibus filiis et filiabus meis, Ferrandus Gonçaluez, Petrus Gonsaluez, Munio Gonsaluez,
Gonsalus Gonsaluez, domna Eluira Gonsaluez//(*fol. 3r*) domna Maria Gonsaluez, Maior Gonsaluez, Vrraca Gonsaluez, non per metum neque perturbatum sensum, sed propriam uoluntatem, dono et concedo pro remedio anime mee atque parentum meorum, tibi, dicto Martino, abbati Sancti Cipriani de Monte Hoca atque omni conuentui, illam albergeriam que uocatur Cernuega cum omni hereditate sua quantumcumque potuimus ganare in Cernuca et in termino de Quintana Suar et de Latio atque ita statuimus ut si aliquis homo ex proienie aut ex genere nostro hanc cartam contradicere uel infringere uoluerit sit maledictus et excomunicatus atque iram Dei omnipotentis ueniat super eum et cum Datan et Abyron quos uiuos terra absorbuit et cum Iuda traditore in inferno habeat partem.

Facta carta XVI kalendas septembris, sub era Mª CCª XXª IIIIª. Regnante rege Aldefonso in Toletula et in Castella et in Burgis. Maiordomus rex Rudericus Guterrez; alfieraz Didac Lupez; merinus maior Lop Diaz; dominante comite Ferdinandus in Asturias et in Aquilar; en Castella et in Borobia Didacus Lopez; en Soria et en Almazan Didacus Semenez; archiepiscopo in Toleto Gonsaluo Petrez; episcopus burgensis Marinus; episcopus palentine Aldericus.

3.1.2.2 Imagen A06v07r

fols. 6v-7r.

1175, febrero, 20. Soria.

(*En tinta roja:)* Priuilegius rex Aldefonsus de Coua Sant.

(*Crismón, alfa y omega*) In nomine Domini amen. Quoniam ad redimenda peccata nichil salubrius helemosinam. Id circo ego Aldefonsus Dei gratia rex Castelle, una cum uxore mea Alienor regina, pro animabus parentum meorum et salute propria, dono et concedo et uobis Gundisaluo Petri de Siones et uxori uestre, Uillam illam que uocatur Coua Sant, in alfoz de Sedano sitam, totam ex integro cum terris uidelicet et uineis, pratis, uineis (*sic*), pascuis, cum montibus et fontibus, cum ingressibus et egressibus et cum omnibus terminis et pertinentiis suis, iure hereditario in perpetuum habendam et possidendam. Si//(*fol. 7r*) quis uero huius mee donationis paginam in aliquo rumperpe (*sic*) uoluerit, iram Dei omnipotentis plenarie incurrat e tregie parti M aureos in cauto persoluat.

Facta carta Sorie, era Mª CCª XIII, Xº kalendas Marcii. Et ego rex A(ldefonsus) regnans in Castella et Toleto, hanc cartam manu propria roboro et et (*sic*) confirmo.

(*Signo rodado*:) Signvm regis Ildefonsi.

(*En círculo alrededor*:) Comes Gundisaluus de Maranones, alferit conf. Rodericus Guterez maiordomus curie regis conf.

(*Primado*:)Cenebrunus, toletanus archiepiscopus et hyspaniarum primas confirmat.

(*1ª columna*:) Petrus, burgensis episcopus conf.; Raimundus, palentinus episcopus. Joscelmus, segontinus episcopus; Comes Nunio conf.; Comes Gomez conf.; Comes Gundisaluus conf.

(*2ª columna:*) Petrus de Arazuri conf.; Albarus Roderici conf.; Petrus Roderici conf.; Didacus Semenet conf.; Gundisaluus Copellinus conf.

Petrus regis notarius, Raimundus existente cancellario scripsit.

3.1.2.3 Imagen A08v09r

fol. 8v.

1229, junio.

(*Al inicio, con la misma tinta*:) La compra de don Moriel.

In nomine Domini. Notum sit omnibus quod ego don Moriel, una cum uxore mea donna Eluira, amos de mancomun, ex nostras bonas uoluntates, uendemus e robramus uobis don Martin de Frias, prior del monasterio de Sancta Maria de Rio secco, e a todel conuento daqueste mismo monasterio e uendemoslo en uuestra uoz a frey Martin de Cisnera e por al conuento de Rio secco toda quanta heredat nos abemos e a nos aperteneçe in Quintana suar, e in suos terminos e quanto compramos e ganamos e todo quanto yo don Moriel herede de mi auola donna Eluira Munnoz e todo quanto compramos de Rodrigo Rodrigez e de Roy Pedrez e de Goncaluo Pedrez, e de nietos de Petro Leon quanto que ellos y auien; scilicet de todo esto; collaços, terras, uineas, casas, solares populatos e non populatos, ortos, molinos, prados, pastos, riuis, aquis, montes e fontes, populato e non populato, con entradas e exidas, totum ab omni integritate. E prendemos de uobis in precio C e LXXX morabedis bonos derecteros e un manto en robra, e sumus de todo paccati de uendida e de robora. Qui ista carta infringere uoluerit abeat iram Dei e in coto regi terri CCCC morabedis persoluat. E ista uendida e ista robora sit uobis don Martin de Frias, prior del monasterio de Sancta Maria de Rio seco e a todel conuento deste mismo logar, duplata e meliorata, in simili e tali loco. Et super hoc ego, don Moriel e ego Martin Alfonso, fijo de don Alfonso Diaz de Roias, amos de mancomum, sumus uos fiadores de sanamiento de ista heredat supra dicta.

Facta carta mense junio, anno ab incarnatione domini Mº CCº XXº nono. Era Mª CCª LXª VIIª. Regnante rege Fredinando, cum uxore sua regina Beatrice in Burgii e in Toleto e in Castella e in omni regno suo. Alfieraz del rey, don Lop Diaz de Faro; maiordomo mayor, don Gonçaluo Roiz; obispo en Burgos, maestre Mauriz; merino mayor del rey, Garci Gonçaluez de Ferrera.

Huius rei testes sunt, de fijos dalgo: Gonçaluo Moriel, Garci Roiz de Mena, Munnio Pedrez de Quintaniella, Roy Gonçaluez de Val dolmiellos, Aluar Petris de Sedano, don Tello dauaias, Ferrant Guterrez de Quintaniella, Diag Alfonso de Quintana ferruz. De Burgos: don Pedro Moro, el alcalde; Rodrigo Antolinez, don Iohannes, yerno de don Petro Uaracon. E es manero de meteruos en la heredat, don Tello dauaias.

Martinus Petri scripsit.

fol. 9r.

1170, junio, 26. Oña.

(*En tinta roja:*) Priuilegius Sancti Stephani.

(*Crismón, alfa y omega*) In nomine domini nostri Ihesu Christi, amen. Ego Ildefonsus Dei gratia rex dono et concedo Deo et uobis domnno Martino, beate Marie de Quintana suuar abbati et monachis uestris presentibus atque futuris, Sanctum Stephanum de Tesla, inter Ualle et Quintanam situm, cum ingressibus et egressibus, et cum omnibus terminis et pertinenciis suis ubicumque inuenire potuerint ut animo liberum ad quietum illud abbeatis et iure hereditario in perpetuum possideatis. Si quis uero ex meo uel alieno hoc meum factum rumpere uoluerit genere, sit a Deo maledictus et excomunicatus et in super regie parti mille aureos persoluat.

Facta carta apud Oniam, era Mª CCVIIIª, VIº kalendarum Iulii. Et ego Ildefonsus rex hanc cartam roboro et confirmo. Cenebrunus Dei gratia Toletane sedis archiepiscopus et yspaniarum primas.

(*Signo rodado*:) Signum regis Ildefonsi.

(*En círculo alrededor*:) Raimundus regis cancellarius conf.; Petrus Garsie maiordomus regis conf.; Rodericus Gonçalui alferit regis conf.

(*1ª columna:*) Petrus burgensis episcopus, conf.; Rodericus naiarensis episcopus, conf.; Comes Nunnio, conf.; Comes Albarus, conf.; Comes Gomez, conf.

(*2ª columna*:) Petrus Roderici conf.; Gonçaluus Roderici conf.; Gomez Martini confirma; Petrus Martinez el mancebo conf.; Gutier Pelaet merinus regis conf.

Petrus notarius regis scripsit.

4. El libro: Evolución Histórica

A lo largo de la historia el libro ha tenido diversos formatos, condicionados por el material empleado para su confección. El primer material utilizado fue la corteza de los árboles. El vocablo "libro", deriva de la voz latina *liber*, es decir, corteza secundaria de los árboles. En la India y en Indochina los libros se escribían en hojas de palma secas y empapadas de aceite, en Asia central en la corteza del abedul, en China en seda y en Roma en telas de lino; también se escribió sobre metal; plata, oro, plomo, en láminas delgadas que se enrollaban como papel. Se sabe que en China hubo una rica producción literaria de alto nivel, acompañada de un gran desarrollo en el arte del libro, que se remonta a tres mil años a. de C. El primer soporte de la escritura fue la madera, sobre la cual se escribía partiendo del ángulo inferior derecho y se continuaba de forma vertical. Después de la destrucción de las tablillas, ordenada por el emperador Qin Shi Huangdi en el año 231 a. de C., se utilizó un nuevo soporte: la seda, sobre la cual se trazaban los signos mediante una caña de bambú, con una tinta compuesta de negro de humo y de goma. En esa misma época florecían otros dos centros de civilización en África Septentrional y en Asia Menor, especialmente en Mesopotamia. Los sumerios y los babilonios fueron quienes adoptaron el sistema de escritura cuneiforme y quienes comenzaron a utilizar las tablillas de de arcilla como soporte. En la ciudad de Nippur, se descubrieron rastros de una gran biblioteca que contenía más de quinientas mil piezas y un archivo de documentos. Al este de Ankara, en la capital de los hititas, se encontraron quince mil tablas de grandes dimensiones con escritura cuneiforme y otras más en Ras-Shamra, en Siria Septentrional.

4.1 El libro manuscrito en forma de rollo

4.1.1 El papiro en Egipto

En Egipto la cultura escrita alcanzó un desarrollo y valor considerables, no sólo en el campo religioso sino también en el científico y literario. El soporte de esa escritura fue la planta de papiro que crecía en el valle del Nilo. El *Cyperus papyrus*, de la familia de las ciperáceas. Los antiguos egipcios, gracias a su nivel de cultura y civilización, descubrieron muy pronto los méritos de tal planta y lo utilizaron para necesidades cotidianas casi durante cuatro mil años. La utilización más importante del papiro en Egipto fue la de ser soporte de escritura. La fabricación de este soporte se realizaba cortando solamente el tallo, se introducía primero en agua, después se le quitaba la corteza verde y se cortaba en tiras de 25 mm de ancho. Las tiras obtenidas se extendían en una superficie plana y se mojaban con agua del Nilo, sobre esta capa se ponía otra en sentido transversal y uniéndolas mediante presión se dejaban secar al sol. Se obtenía así una hoja compacta que se aplanaba con un martillo, se pulía y alisaba con un instrumento de marfil, después se cortaba para obtener hojas de un mismo formato, entre 12 y 13 cm de largo y de 22 a 33 cm de alto, finalmente las hojas se envolvían con forma de rollo y algunos se comercializaban. La tonalidad más blanca y delgada de los rollos estaba destinada a la escritura de los libros sagrados y la tonalidad más oscura a los de tipo ordinario.

El libro egipcio tenía forma de rollo formado generalmente por 20 hojas enrolladas alrededor de una varilla de madera, hueso o marfil. La escritura se trazaba paralelamente a las fibras horizontales, por lo general solo en una de las caras de la hoja, formando columnas estrechas que se numeraban progresivamente.

Los papiros "opistográficos" estaban impresos en las dos caras. Las columnas se llamaban *paginae* y también *schedae*, la primera hoja se conocía como "*protocollo*", la última por "*excatocollo*". La longitud del rollo variaba según la necesidad, la escritura utilizada en los papiros no era igual a la de los jeroglíficos de las inscripciones, sino que más bien presentaba una forma más rápida y fácil de comprender, se llamaba **hierática** (sacerdotal). Solo en épocas sucesivas se utilizó una escritura más cursiva llamada **demótica** (popular).

Los escribas egipcios utilizaban para escribir una varilla de bambú cortada transversalmente que girada en diferentes sentidos, pudiendo formar trazos gruesos o finos. La tinta se preparaba con hollín o carbón vegetal, extraído de los utensilios de cocina y tratado con una ligera solución de cola. La tinta roja se utilizaba para los títulos y los comienzos de capítulo.

El rollo de papiro se conservaba en una especie de recipiente de madera o de arcilla. La mayor parte de los papiros que se han conservado ha sido gracias a la costumbre religiosa según la cual debían de depositarse en la tumba. A pesar de la aceptable calidad del papiro como soporte de escritura, se degrada rápidamente cuando está expuesto a un alto grado de humedad. Por este motivo, prácticamente la totalidad de los papiros conservados se encontraban en tumbas. La mayor parte de los mismos son los llamados "libros de los muertos"**.** Estos libros eran rollos de papiro realizados por sacerdotes que los hacían dejando en blanco solo el nombre del fallecido y los adornaban con dibujos más o menos elaborados según la categoría del destinatario, también se encargaban de su venta siendo esta la única forma de comercio de libros conocida en el antiguo Egipto.

Por otro lado las bibliotecas egipcias estaban asociadas a los templos. En el sur de Egipto, en Edfu, se descubrió en el templo de Horus, dios del sol, una sala cuyos muros se habían adornado con los títulos de los libros conservados en su biblioteca, además cerca de Tebas se han encontrado dos tumbas cuyas inscripciones mencionan la calidad del "bibliotecario".

4.1.2 El papiro en la antigüedad grecorromana

El papiro hizo su aparición en Grecia hacia el siglo VII a. de C., época en que nació la poesía lírica. Los griegos llamaron a una hoja de papiro aún no escrita *charta*, en latín y en italiano "carta"; al rollo se le llamo en latín *volumen* o *liber*. El papiro griego más antiguo es del siglo IV a. de C. Una mayor cantidad apareció en las excavaciones realizadas a finales del 1.800 en Egipto y Asia Menor; éstos pertenecían al siglo III a de C. Con Ptolomeo I que, después de la caída del Imperio de Alejandro, consolidó su poder en el valle del Nilo e hizo de la capital, Alejandría, una de las ciudades más civilizadas y de mayor entidad cultural. Su hijo Ptolomeo II fue el impulsor de la renombrada biblioteca de Alejandría. Se decía que la biblioteca conservaba alrededor de setecientos mil rollos de papiro. El texto de las obras se distribuía en varios rollos, de una longitud más o menos similar, teniendo en cuenta la división por capítulos, mientras que los textos breves estaban reunidos en un mismo rollo. Esto indica una cierta tendencia a adoptar materiales de la misma longitud; alrededor de seis o siete metros, que formasen un cilindro de unos cinco o seis centímetros de diámetro, cómodo para llevar en la mano. En cuanto a la altura, ésta variaba entre doce y quince centímetros, o bien entre veinte y treinta. La escritura trazada sobre el papiro apareció aumentada, sin espacios entre una palabra y otra; sin embargo se señalaba el final de un párrafo subrayando la última línea de este. El título de la obra se comenzó a utilizar muy tarde, generalmente se citaba al final del texto. En Grecia floreció un importante y bien organizado arte librario, cuyos productos también se exportaban al extranjero. El copista y el vendedor de libros al principio fueron una misma persona; solamente a partir del siglo V a de C., los comerciantes llamados "*bibliopoli*", formaron un gremio independiente que realizaba su trabajo en negocios abiertos al público, el local además de ser punto de venta, fue lugar de encuentro de personas eruditas que se reunían para escuchar la lectura en voz alta, estas personas tenían la tarea de la difusión del libro.

En Atenas, el tirano Pisístrates tuvo el mérito de promover en el año 550 a. de C. una biblioteca pública. También fue conocida la biblioteca privada de Aristóteles que, a su muerte, pasó a la célebre biblioteca de Alejandría en Egipto. En Roma la utilización del papiro como soporte de la escritura era bastante más cómoda y fácil de manejar que la corteza de árbol. Empleaban rollos de plomo y de tela. Los volúmenes destinados al comercio estaban escritos por esclavos *literati*, llamados *scriptores*, *amanuenses*, *librarii*, *antiquarii*. *Librarius* significaba "escritor de obras literarias". La calle de los libreros de Roma se llamaba Argileto y estaba situada cerca del teatro Marcello. En Grecia al igual que en Roma, se hacían reuniones periódicas de literatos y gramáticos en estas librerías donde leían fragmentos de sus obras a los críticos y al público.

Por otro lado para conseguir copias más correctas, y también porque el comercio de libros se limitaba a las obras más buscadas, los romanos amantes del estudio tenían en sus casas esclavos *literati* encargados de copiar textos. En Roma eran miles los esclavos que se dedicaban a la transcripción. La pasión por los libros trajo como consecuencia la formación de bibliotecas privadas, además de las públicas, instruidas por César y Augusto. El comercio de libros, que al principio ofrecía volúmenes a altísimos precios, fue reduciéndolos a medida que la producción aumentaba. Los libreros romanos tenían un catálogo de las obras en venta, con el nombre del autor y las primeras palabras del texto, los volúmenes se conservaban sobre pequeños palcos llamados "nidos" y protegidos a menudo, por un paño color púrpura. En Roma surgieron fábricas que importaban de Egipto papiro no elaborado, que se transformaba en

fardos de hojas preparadas para ser escritas. Los Ptolomeos, para proteger su producto, lo gravaron con un arancel de exportación y más tarde monopolizaron su comercio sellando la primera hoja de cada fardo con una especie de timbre oficial llamado "protocolo".

La conquista de Egipto por los árabes en el año 641, redujo el ritmo de exportaciones que llegó a ser muy irregular, seguramente fueron factores económicos y sociales los que provocaron la desaparición del papiro. Primero, el alto costo del transporte por barco, segundo, la sequía del Nilo y los trabajos de irrigación comenzados por los árabes, que transformaron los pantanos en terrenos agrícolas.

Los árabes también introdujeron al cultivo del papiro en Sicilia, sobre todo en la zona de Siracusa, donde aún hoy florece frondosamente. En la Edad Media, el papiro era conocido también con el nombre de "*paperio*" o "*pampero*" y por similitud "*parruca*" o "*piluca*". Siglos después en Siracusa, en 1780, el naturalista Saveric Landolina redescubrió la planta de papiro cerca del río Anapo. Hacia el final del siglo X, también se tiene noticias del papiro en Palermo y por lo tanto puede ser que los papiros más recientes de Rávena y Roma hubiesen sido producidos en Sicilia.

La utilización del papiro fue prácticamente universal hasta finales del siglo III, cuando empezó a ser sustituido por el pergamino.

4.2. El libro manuscrito en forma de códice

La palabra ***códice*** tiene un doble significado. Por un lado, en cuanto a la morfología libraría, designa al libro de páginas, es decir, el que está compuesto por un conjunto de cuadernos formados al doblar una o más hojas del soporte de que se trate, cuadernos que luego se cosen y se encuadernan. Por otro lado, en cuanto a la tecnología de escritura, el término ***códice*** se usa para diferenciar el libro de páginas manuscrito del impreso.

4.2.1 Las tablillas enceradas

En el mundo grecorromano el papiro no fue el único material utilizado como soporte de escritura, se emplearon también las "tablillas enceradas". Estas estaban formadas por tablillas rectangulares de madera, con un pequeño margen realzado a lo largo de los cuatro lados, en la parte central que estaba hundida se esparcía la cera y sobre ella se escribía con un instrumento puntiagudo, el *stilus*, en la parte opuesta a la punta tenía un rascador, de forma que se pudiese borrar fácilmente la escritura rascando la cera. Las tablillas enceradas se utilizaban para ejercicios escolares, para cuentas, comunicaciones epistolares y borradores de los poetas. Generalmente se unían con cuerdas formando una especie de libro, según él número se obtenía un díptico, un tríptico, o un políptico. Los documentos dípticos o trípticos, se presentaban con el texto en doble redacción.

El grupo más importante lo componen las tablillas descubiertas el año 1875 en Pompeya, en la casa del banquero L. Cecilio Giocondo, son 127 y comprenden desde el año 15 al 62 d. de C.; estas contienen los recibos de la administración municipal, de la cual Giocondo era el arrendador.

Un soporte parecido a las tablillas enceradas serían los dípticos consulares de marfil ricamente esculpidos. Son aproximadamente 71, de los cuales el más antiguo es un díptico sacerdotal del año 388. En la Edad Media se utilizaron para usos litúrgicos como tapas de evangelios y misales de gran lujo.

4.2.2 El códice

El rollo o volumen, que fue la primera forma del libro en la civilización antigua del mundo occidental y en Oriente, entró en competencia con el códice al principio de la era cristiana y posteriormente fue sustituido por este. La etimología de la palabra es *caudex*, tronco de árbol o corteza. El destino del códice fue más brillante que el del rollo. Son numerosos los interrogantes respecto a la técnica con la que los artesanos medievales confeccionaron el libro manuscrito bajo forma de códice con soporte de pergamino.

4.2.2.1 Los soportes de libro en formato códice: El pergamino

Según la tradición, se atribuye a la biblioteca del rey de Pérgamo al mérito de haber convertido en uso público la utilización del pergamino como soporte de escritura. Ya desde tiempos antiguos se había utilizado el cuero para este fin. Pero solamente alrededor del siglo III a de C., se inició un nuevo tratamiento del cuero, de forma que se adoptase mejor para recibir la escritura, tal innovación sucedió en Pérgamo, por lo tanto el pergamino es un "papel" de piel animal convertida en hojas aplanadas y lisas que permitían su utilización óptima como material de escritura.

Para la preparación de pergaminos se utilizaban pieles de animales de oveja, cabra, cordero y ternera; en Egipto se empleaban pieles de antílope o de gacela para obtener pergaminos de mejor calidad. Para su utilización, los pergaminos se purgaban introduciéndolos durante unos días en cal y mientras era flexible, se afeitaba por las dos partes para eliminar la grasa y quitarle las manchas, después se pulía con piedra pómez y se reducía al tamaño deseado. El pergamino destinado a los códices era más fino y pulido, dado que se utilizaba por los dos lados, mientras que el de los documentos se pulía por un sólo lado.

Los romanos acostumbraron a teñir los pergaminos de amarillo o de rojo, aparentemente porque su blancura se ensuciaba fácilmente. Para los códices de lujo se utilizó el color púrpura, con escritura de oro y plata, el más famoso fue el Codex Argenteus, llamado de Ulfila, porque representaba los Evangelios traducidos a lengua gótica por el obispo Ulfila, escritos con letras de plata de forma uncial.

Entre los tintes dados al pergamino en la época humanística ocupó el primer lugar el púrpura, mientras que los casos de pergamino coloreados con azafrán o en negro fueron muy raros o limitados. En la Alta Edad Media sé re utilizaban frecuentemente los pergaminos ya escritos para nuevos códices. Con este fin se borraba la escritura sumergiéndolos en leche y restregándolos con piedra pómez, son los llamados palimpsestos o *codices rescripti*. Se borraban igualmente textos profanos y sagrados, ya que a menudo estaban estropeados, mutilados o fuera de uso por su antigüedad. Hubo manuscritos reescritos dos o más veces. Más tarde se consolidó también la costumbre de utilizar el pergamino solamente para las actas de ciertas autoridades, como Papas o altos funcionarios.

4.2.2.2 Los soportes de libro en formato códice: El papel

El papel tuvo su origen en Oriente; el descubrimiento se le atribuye a China y precisamente al director de los talleres imperiales, Ts' ai Lun, que al principio del siglo II d. de C. tuvo la idea de fabricar una especie de pasta delgada sacada de la corteza de la morena, del cáñamo y de material de desecho de tela o seda. Para fabricar papel de lujo se utilizaban trapos de cáñamo, algodón y lino que al golpearlos con mazos se obtenía una pasta liquida y homogénea, posteriormente se obtenían delgados hilos; un cuadro móvil determinaba el espesor del papel, se procedía a su secado con fieltros y exponiéndolo al aire se realizaban las operaciones de encolado y satinado.

Al final del siglo XVII los holandeses comenzaron a utilizar un sistema de trituración de los trapos usados mediante cilindros y laminillas; en la segunda mitad del siglo XVIII James Whatman, en Birmingham y los Montgolfier, en Annonay, consiguieron papel de seda sin huellas de los alambres. Sin embargo, no fue sino hasta 1844, cuando Frederich Keller inventará la industria de papel de madera. Una particularidad del papel son las filigranas o marcas de fábrica es decir los distintivos del fabricante que aparecen en el papel, tienen forma de letras, figuras de animales, flores, frutas, arneses y utensilios dibujados con hilo de latón o de plata.

4.2.3 La estructura y la composición

A partir del siglo IV, al prevalecer definitivamente el pergamino sobre el papiro, el *codex* sustituyó al *volumen* y desde entonces ha constituido la forma habitual del libro. Estructuralmente la unidad mínima de composición del códice era el bifolio, es decir, una pieza de pergamino doblada sobre sí misma. Los bifolios se encajaban unos dentro de otros para formar cuadernos, siendo los más habituales los de cuatro bifolios (“llamados cuaterniones”, de donde derivará precisamente el término “cuaderno”, y compuestos por tanto por 8 folios y 16 páginas).

El códice medieval no tenía portada, comenzaba con una fórmula de *incipit* (en latín, “comienza”), en la que se anunciaba el título de la obra. Dicha página podía estar adornada con tintas de color y motivos ornamentales variados (vegetales, geométricos y arquitectónicos). El

nombre del autor figuraba a veces al final de la obra, en el colofón, pero muchas veces, hasta bien avanzada la Edad Media, se trata de obras anónimas.

Cuando el escriba había acabado su trabajo, empezaba el del rubricador, que escribía en tinta roja los títulos de los capítulos y epígrafes, y adornaba las letras iniciales de las frases con un trazo vertical. Para asegurar la regularidad de la escritura y la armonía de la página se trazaban en el pergamino líneas horizontales y se delineaban los márgenes con dos líneas verticales en seco con una punta de metal.

Al final del siglo XII se empleó también tinta negra para trazar líneas. Dado que el número de cuadernos que componían un manuscrito podía proceder de una fuente en desorden, los copistas procedieron a numerar los cuadernos en la última página. El procedimiento se conoce como "*signatura*", también se numeró cada hoja con la letra del pliego al que pertenecía seguida del número progresivo de cada pliego.

4.2.4 Los instrumentos para escribir

La tinta empleada estaba compuesta por diferentes ingredientes, en la Edad Media entraron a formar parte de la composición de la tinta elementos como vitriolo, agalla de encina, cerveza y vinagre. En la época carolingia se empezaron a utilizar tintas con matices rojizos. Para la escritura en general la tinta negra es la que se utilizaba más a menudo.

A partir del siglo XII, además del rojo se utilizó para las iniciales el azul y el verde. Este último sobretodo en documentos orientales. La escritura en oro y plata sobre fondo púrpura es de origen bizantino.

Los códices áureos y plateados pertenecen al siglo IX; en la época carolingia se hicieron varios manuscritos en pergamino teñido en negro, con los títulos, las iniciales y el nombre de Dios en oro y plata.

Como instrumentos para la escritura sin tinta se empleaban el ya citado *stilus*, para tablillas enceradas, y la pluma de hierro. Su utilización debió ser muy antigua ya que se habla de ella en la Biblia. Se utilizaba normalmente para escribir sobre muros, terracotas blandas y hasta metales.

Para las escrituras con tinta se empleaba el cálamo o caña de junco, hasta que fue sustituido por la pluma de ave, especialmente de oca. El pincel se utilizó más raramente y sobre todo para la escritura en oro y los códices. El "lápiz" fue adoptado mucho más tarde, cuando se descubrieron los yacimientos de grafito, llamado *plombagina*, sin embargo el lápiz sólo fue de uso común a partir del año 1700.

4.2.5 Abreviaturas

Un capítulo aparte de la evolución de la escritura lo constituyen las abreviaturas. Según los periodos y las formas de la escritura, éstas se utilizaron para aligerar lo extenso de la obra y para reducir el consumo de soporte escritorio. Los sistemas de abreviatura fueron dos: el de representar

la palabra con un significado convencional y el de escribir el vocablo con una o más letras. En este segundo caso se pueden emplear dos sistemas: el de *contracción*, si se mantienen las primeras y las últimas letras, omitiendo las intermedias; y *suspensión*, cuando se conserva la parte inicial de la palabra y se suprime el final (las *siglas* serían el caso más extremo).

Para indicar la existencia de abreviaturas se emplearon signos de abreviación. Los signos generales de abreviación fueron sobre todo tres: el punto, la línea pequeña o lineta y la letra pequeña sobrescrita. Estos signos se utilizaron con significado "general" o "indeterminado". El punto es el signo de abreviatura más antiguo. Asímismo, como signo de abreviatura empezó a aparecer el signo I, su desarrollo aumentó cuando se introdujo la contracción. Las abreviaturas se utilizaban para escribir más rápidamente y ganar espacio en un soporte no demasiado económico, como el pergamino o la vitela, no obstante, la proliferación de abreviaturas, reduce de forma considerable, la legibilidad del manuscrito, dificultando el reconocimiento de sus caracteres.

Al difundirse la escritura minúscula se tuvo la necesidad de crear nuevos signos y en la época carolingia, con su perfeccionamiento, se puso mayor cuidado a la hora de puntuar. Durante el siglo IX, a menudo aparecía un punto en la mitad de la línea para una pausa breve; un punto seguido de una coma; un punto sobre una coma; dos puntos sobre una coma para pausa final. En las reglas del *ars dictandi* se indicaba, punto y barra para la pausa breve; punto o dos puntos para la pausa media; punto y coma para la pausa final.

El punto interrogativo se presentó de diferentes formas, la más conocida fue un punto sobre el que se trazaba una línea curva oblicua. Los acentos se trazaban como ápices sobre monosílabos largos o como signos tónicos.

Hasta el siglo XVI la numeración utilizada en los manuscritos latinos era la romana; más tarde también se adoptaron las cifras arábigas. Estas últimas, de origen hindú, representaban las letras iniciales del nombre de los números en sánscrito. Los primeros ejemplos de números árabes en códices latinos se dieron en España, en dos manuscritos de la Biblioteca del Escorial, del 976 y 992.

4.2.6 La decoración y la ilustración.

La importancia del libro ilustrado fue tal que sirvió como fuente de inspiración para otros sectores artísticos como tapices, esculturas, vidrieras, y más tarde también para esmaltes. Los artistas que se dedicaron a la decoración del libro demostraron el peso que la miniatura tuvo no solo desde el punto de vista artístico sino también a nivel pedagógico y su papel en el lenguaje de la escritura.

El término "miniar" significa "colorear en rojo" deriva de la palabra *minium*, con la que en la Edad Media se solía llamar al cinabrio, es decir, el sulfuro de mercurio de color rojo vivo que se encuentra en abundancia en la naturaleza como mineral de mercurio. La interpretación más sencilla parece la de "dar alumbre", es decir dibujar con lacas alumbradas obtenidas por la reacción química del alumbre de roca con algunas materias colorantes vegetales. Para poder escribir o miniar sobre pergaminos blancos o coloreados, primero eran tratados para que fuera más fácil la aplicación de tintas y colores. Se aplicaba sobre la superficie a decorar, bilis de buey mezclada con clara de huevo o se frotaba un algodón empapado con una solución diluida de cola con miel. Los pinceles para miniar se hacían con pelos de cola de marta cibelina, introduciéndolos en un canutillo de pluma de oca de gallina o de paloma.

El esbozo del dibujo se hacía con un lápiz de plomo. Para borrar se utilizaba la miga de pan. También se empleaban una serie de cuchillos con varias hojas para sacar punta a las plumas y cortar el pergamino. Otros utensilios eran la escuadra, la regla y los tinteros de tinta negra y roja, algunas veces se utilizaba el compás.

Para aclarar los líquidos o separar los colores de las soluciones depurativas se colaban en un filtro cónico. Los morteros para hacer mezclas eran de mármol calcáreo. Para ligar colores se utilizó la goma arábiga y la clara de huevo; se empleaban mezclas como soluciones de albúmina, azúcar, miel y clara de huevo. La conservación de estas soluciones de goma y colas se aseguraba añadiendo algún aséptico como el alcanfor, clavos de clavel, vinagre o jugo de ajo.

El alumbre de roca y el de azúcar se empleaban para obtener barnices coloreados a partir de extractos vegetales. Cada color utilizado para las miniaturas en la Edad Media tuvo características diferentes; por ejemplo, el azul ultramar se obtenía moliendo la piedra lapislázuli y se extraía de las minas de Badaksham.

Hasta los siglos XII y XIII se dibujaba con lapislázuli molido y lavado; después se hicieron purificaciones consistentes en empastar el polvo mineral con ceras, aceites y resinas hasta conseguir una pasta maleable. La madera roja de Oriente se utilizó mucho en la Edad Media para teñir fibras, y también para preparar lacas rojas alumbradas para miniar.

Una vez acabado el trabajo del copista, el miniador dibujaba sobre los pergaminos, el esbozo de figuras y ornamentos con el "lápiz de plomo", trazando también, además de los principales contornos, las líneas de los pliegues de los vestidos y los límites de las zonas de sombra y de luz. Una vez acabado el esbozo, y si se había previsto su uso, se extendía el mordente, es decir, se aplicaba una película dorada. Este procedimiento se llamaba *doratura* (pan de oro auténtico o no), se lijaba y se bruñía. Al principio se utilizaba más un campo de fondo monocromo o también

dorado. La delicadeza y transparencia típicas de la miniatura se definieron con la aplicación de la gama de colores y por un empaste más completo.

La elaboración de las miniaturas llegó a ser más sofisticada y compleja en la Baja Edad Media. Una vez terminada la obra, el ilustrador daba un último barniz con goma arábiga y clara de huevo que imprimía al trabajo una pátina brillante.

Al principio este arte fue practicado únicamente por el clero, ya que existió solo en los monasterios. En la ilustración de un manuscrito participaban varios colaboradores según las tareas asignadas; el miniaturista hacia las partes secundarias como las letras, adornos, bordes y decoración y el pintor, se encargaba del trabajo más importante, es decir, la ilustración. El miniaturista ejecutaba las condiciones escritas con anterioridad en el códice o con esbozos que indicaban las líneas generales de la escena a representar.

El manuscrito medieval de Occidente adoptó tres elementos diferentes de decoración, la inicial, el borde y la ilustración miniada. Se perfilaba el cuerpo de las letras, y alrededor de ellas se entrelazaban tallos y hojas con cabezas de animales estilizados y figuras fantásticas prevaleciendo el rojo y el verde.

Los manuscritos franceses tenían una decoración rica en colorido que se desarrolló todavía más con la difusión de la minúscula Carolingia, a la miniatura carolingia sucedió la otoniana, producida por escuelas como las de Reichenau, Einsiedeln o Echternach, esta miniatura se creó en Inglaterra y más tarde paso a Irlanda dando vida al estilo anglo-irlandés.

Como expresión de escuela miniaturista local de los siglos X – XII se pueden citar la miniatura mozárabe española, con intensos amarillos, rojos, azules y verdes y un intenso efecto decorativo en los bordes extravagantes y la vitalidad popular de figuras y animales en escenas alegóricas - trascendentales. Hasta el siglo XII, la miniatura fue exclusivamente monástica pero, en el siglo XIII, con el desarrollo de las ciudades y la aparición de las universidades, llegó también al mundo laico y se utilizó en estatutos, obras literarias y romances de caballería. La miniatura llegó a su apogeo en el siglo XV, teniendo en Italia su mayor esplendor. La aparición del libro impreso significó el final de la utilización de la miniatura para códices.

4.2.7 La encuadernación

En el siglo I d.C. los cuadernos que contenían varias hojas de papiro o pergamino se insertaban entre dos tablas de madera o de hojas de papiro encoladas entre sí. En Occidente, las encuadernaciones más antiguas que se conservan se remontan al siglo VII (las tapas del Evangelio de la reina Teodolinda de Monza, compuestas por una plancha muy fina de oro con ocho camafeos dispuestos en forma de cruz); en ellas los pliegos van cosidos mediante el procedimiento de doble nervadura y se unían a la cubierta más tarde. Las dos tapas se sujetaron después del cosido. Una vez fijadas las tapas a los

pliegos, podían revestirse de diferentes formas.

Los textos sagrados se decoraban con oro, piedras preciosas esmaltes y marfil. En dichas encuadernaciones pueden distinguirse cinco épocas:

La *bizantina*, desde el siglo IV al VII, en que las tapas se cubrían de oro o plata con pedrería según los modelos preciosos de Constantinopla. A este periodo corresponde el evangeliario del Tesoro de Monza, debido a la reina Teodolinda, siglo XV.

La *prerrománica de Occidente*, desde el siglo VII al XI en que las tapas se exornaban con marfil labrado, costumbre que se había iniciado en el siglo V. A esta época corresponde el Misal de la biblioteca Barberini, en Florencia siglo VIII.

La *románica*, durante los siglos XI y XII en que estas láminas de marfil encuadraban normalmente o se engastaban en un marco de plata u oro con pedrería. De este tiempo datan los marfiles y tapas de la catedral de Jaca y Museo Episcopal de Vic siglos XI y XII.

La *gótica*, en los siglos XIII y XIV que se distingue por la desaparición del marfil y por el uso de plata repujada y algunas piedras finas. De este periodo es el Evangeliario de la colegiata de Roncesvalles, sobre el cual juraban los reyes de Navarra, y las tapas de otro que guarda el museo vicense.

Por último, podemos distinguir el *final de la época gótica* y el comienzo del *Renacimiento*, donde se emplean indistintamente la plata, los guadameciles (cuero trabajado), las maderas labradas y los terciopelos. Hacia 1470 se introdujo en Italia el dorado en caliente con pan de oro, técnica aprendida por los obreros sarracenos emigrados de Siria y Egipto, abriendo una nueva vía a la ornamentación de la encuadernación, este motivo decorativo se desarrolló en Venecia, bajo el influjo combinado del renacimiento y el Arte bizantino y el oriental.

5. Evolución Histórica de las Escrituras Medievales

Para fijar las periodizaciones de la escritura, es decir, para dividir en periodos el ámbito cronológico expuesto, los tratadistas han seguido diversos criterios cronológicos, morfológicos, geográficos, etc. Cualquiera de ellos, por sí solo, no tiene validez plena ya que una combinación acertada de los mismos es la única que puede dar lugar a una clasificación eficaz y operativa. Por este motivo, hemos decidido seguir el criterio de autores como Steffens [1929], Centcetti [1956] y Bischoff [1958] entre otros, que, basándose en criterios gráficos, distribuyen la escritura en seis periodos o ciclos paleográficos:

1. Periodo de la escritura de la edad romana y de unidad escrituraria, siglos VII a.C. al V a.C.
2. Periodo de las escrituras precarolinas o nacionales, siglos V al VII.
3. Periodo carolingio, siglos VIII al XI.
4. Periodo gótico, siglos XII al XIV-XV.
5. Periodo humanístico, siglos XV al XVII-XVIII.
6. Periodo contemporáneo, siglos XVII ex. al XXI.

5.1 Periodo romano

Durante esta época histórica se pueden diferenciar dos etapas: La antigua escritura romana y la nueva escritura romana.

A la antigua escritura romana, siglos VII a.C. al II a.C., se la llama también “capital” o “mayúscula”, en honor al módulo de sus letras y presenta dos tipos bien caracterizados: el “arcaico” y el “clásico”.

La denominada escritura “arcaica” o “lapidarioa” es sin duda la más antigua y todos los objetos escritos en estos caracteres están trazados sobre piedra o material duro. Suelen citarse como testimonios paradigmáticos el “Lapis niger” y la “Fíbula praenestina”.

Idéntico origen y procedencia tiene la denominada “clásica”, datada y de uso frecuente en el siglo II a.C. al IV d.C. Recibe este nombre por haber servido de principal vehículo y medio transmisor de la cultura clásica.

Su trazado admite, en un primer grupo, dos formas o modelos principales, designados con los nombres de escritura “cuadrada” o “elegante” y escritura “rústica” o “espontánea”. Estos modelos se empleaban más en inscripciones que en códices. Aunque su periodo áureo y su utilización más habitual se circunscriban a los siglos II a.C. al V d.C., pueden darse de manera excepcional, en las centurias siguientes.

En la actualidad, los especialistas prefieren hablar y diferenciar la escritura “capital epigráfica” de la “capital libraría”, según esté trazada sobre material duro o sobre material blando, en forma de rollo o de códice.

La “epigráfica monumental” se consolida durante la época del emperador Augusto, con un “ductus” pausado, en sistema bilineal de renglón y con la proporción de las letras inscrita en un paralelogramo. La “capital libraría”, a su vez, se divide en los dos modelos citados: Cuadrada y rústica.

La escritura *cuadrada*, presenta un tipo de letra de módulo cuadrado y realizado de forma muy caligráfica, que reproduce las normas de la epigráfica en el ámbito librario.

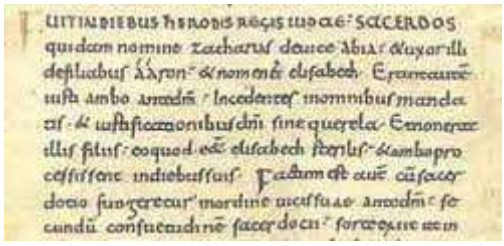
Por su parte, la escritura *rústica*, también es una escritura cuidada, correcta y bastante elegante, pero no es tan vertical ni tan simétrica.

El segundo grupo corresponde a la escritura *cursiva*, también llamada “común”, por ser la empleada en los asuntos más ordinarios y corrientes, como por ejemplo la correspondencia epistolar, escritos de contabilidad, documentos administrativos, etc. Se empieza a utilizar desde el siglo I y llega al IV de nuestra era. Se encuentra trazada sobre papiros, tablillas enceradas y material semiduro. De ella deriva un subconjunto que se denomina “Litterae coelestes” o “imperial” siglos IV al V, ya que su empleo estaba restringido a la cancillería imperial.

La segunda etapa corresponde a la escritura *nueva romana*, denominación establecida en contraposición a “antigua”. Por esta misma razón se la llama también “minúscula” y “moderna”. A partir del siglo IV se generaliza como escritura tradicional del mundo latino.

5.2 Periodo de las escrituras precarolinas

Las escrituras precarolinas, hasta hace poco llamadas comúnmente “nacionales”, conforman un grupo de escrituras provenientes de la “nueva romana” en su versión común, que aparecen a partir del siglo V. La denominación de “nacionales” se utilizó, en principio, para designar a las escritura trazadas por cada uno de los pueblos o naciones bárbaras que se asentaron sobre lo que habían sido territorios de Imperio Romano. Por su parte, el término “precarolinas”, apunta a u proceso gráfico que comenzó en la escritura romana y terminó con la canonización de la carolina. Pertenecen a este periodo las siguientes escrituras:



Merovingia: Su nombre está vinculado a Meroveo, fundador de la monarquía franca en las Galias, aunque también se la llama “franca” o “gálica”. Su periodo de empleo se extiende desde el siglo VI al IX. Es una escritura por lo general, difícil de leer, el especial la cursiva. Son célebres los documentos reales merovingios de la abadía de Saint Denis.

Cursivas italianas: Grupo de escrituras empleadas en territorio italiano, entre las que sobresalen la curial romana, la raventense, la napolitana y la longobarda. Esta última es la principal.

Beneventana: Esta denominación guarda estrecha relación con el ducado de Benevento, en el sur de Italia, en la antigua Dalmacia y en la región de Bari. Se trata de una escritura muy correcta, empleada tanto para códices como para documentos.

Visigótica: Bautizada así en referencia al pueblo visigodo, asentado en l Península Ibérica desde el siglo V al VIII. Su vigencia es desde el siglo VII al XIII. Muestra dos vertientes claramente definidas: redonda o sentada libraria y cursiva, característica de documentos.

Insular: Esta terminología proviene etimológicamente de "ínsula" y resulta muy apropiada para designar la escritura propia de las Islas Británicas e Irlanda. Al contrario que las anteriores, este modelo escriptorio tiene su génesis en pueblos que, sin moverse de su territorio, reciben por diferente caminos, la cultura latina en sus diversas manifestaciones, entre ellas la propia escritura. En estos territorios, la principal vía de penetración de la "nueva romana", de la cual procede, será el cristianismo. Podemos distinguir dos modalidades, la anglosajona y la irlandesa.

5.3 Periodo carolingio:

La escritura carolina o carolingia, con fundamentación gráfica en la "nueva romana", surge como rechazo a las escrituras "nacionales", demasiado cursivitizadas, en especial contra la merovingia. Su procedencia está marcada por la ceración del imperio de Carlomagno (Karolus), de quien se toma el nombre y la cultura que floreció en su entorno. Se utiliza desde finales del siglo VIII hasta comienzos del XIII, aunque algunos especialistas prefieren denominar "precarolina" la de de las postrimerías de la centuria octava. La letra carolina sirve de base para todas las escrituras que existieron después, incluso para la imprenta.

5.4 Periodo gótico:

Se denomina escritura "gótica" a la sucesora de la carolina, de la que proviene directamente. Al igual que su progenitora, es una escritura internacional. Varias son las causas y motivos por los que surge; entre otros, la creación de los Estudios Generales, la industrialización de la escritura, la secularización de la cultura y el predomino del papel como soporte escriptorio en detrimento del pergamino. Su perfil anguloso es característico.

En una primera división de este modelo escrituario, se puede diferenciar una gótica "libraría" y una gótica "documental". La letra libraría corresponde con la caligrafía empleada en manuscritos lujosos y sobre todo en libros litúrgicos, en los que domina las grandes dimensiones, la regularidad y la simetría, que generalmente reciben los nombres de "textura" o "letra de texto". A veces, las formas de la textura aparecen artificiosamente decoradas por medio de la descomposición de las líneas rectas verticales arriba y abajo o solo arriba, dando lugar a una característica formación de una especie de cuadrángulos, por lo que los maestros caligráfico la denominaron "textus fractus", si están arriba y abajo y "textus semifractus", se están sólo arriba.

En el otro extremo se encuentran aquellas escrituras librarías pero poco caligráficas, con las que dse confeccionaban los manuscritos más corrientes, de los contenidos más variados, muchos de ellos en lenguas vernáculas. Son las llamadas "litterae bastardae, aparecidas entre el 1200 y 1300 para ciertos usos librarios subordinados y en cierto modo alternativos.

El último tipo de las escrituras librarías reúne a aquellas que se agrupan bajo las expresiones "letterae scholasticae" o "escrituras universitarias, que en términos de cursividad ocupan un lugar intermedio entre los dos primeros grupos. Son características de los grandes centros universitarios europeos.

Las escrituras contenidas en el códice del AHN que se han usado como banco de prueba para este proyecto corresponden a este periodo gótico

5.5 Periodo humanístico:

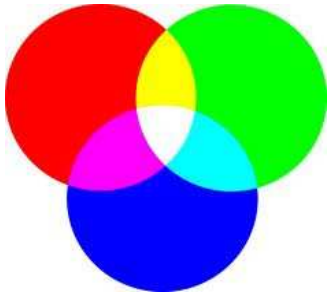
El nombre de esta escritura proviene de los humanistas italianos, que la introdujeron y la aplicaron en la práctica. Surge en la península italiana en el siglo XV, no obstante, había sido propugnada con anterioridad por otros humanistas, como Petrarca y Boccaccio. También se denomina renacentista y su procedencia hay que encontrarla en la letra carolingia.

5.6 Periodo contemporáneo:

En la etapa contemporánea, desde segunda mitad siglo XVIII a nuestros días, es más apropiado diferenciar las escrituras atendiendo a su nacionalidad o su país originario, que a la propia grafía, aunque su génesis siempre está en la escritura humanística cursiva. De esto modo se puede hablar de letras "bastardillas".

6. Imágenes digitales empleadas y sus características

6.1 Imágenes JPEG



JPEG además de ser un método de compresión, es a menudo considerado como un formato de archivo. JPEG/Exif es el formato de imagen más común utilizado por las cámaras fotográficas digitales y otros dispositivos de captura de imagen, junto con JPEG/JFIF, que también es otro formato para el almacenamiento y la transmisión de imágenes fotográficas en la World Wide Web. JPEG/JFIF es el formato más utilizado para almacenar y transmitir archivos de fotos en Internet. Estas variaciones de formatos a menudo no se distinguen, y se llaman JPEG.

6.1.1 Compresión del JPEG

Es un algoritmo de compresión con pérdida. Esto significa que al descomprimir la imagen no obtenemos exactamente la misma imagen que teníamos antes de la compresión.

Una de las características que hacen muy flexible el JPEG es el poder ajustar el grado de compresión. Si especificamos una compresión muy alta se perderá una cantidad significativa de calidad, pero obtendremos archivos de pequeño tamaño. Con una tasa de compresión baja sucederá lo contrario, obtenemos una calidad muy parecida a la del original, y un archivo mayor.
Esta pérdida de calidad se acumula. Esto significa que si comprime una imagen y la descomprime obtendrá una calidad de imagen, pero si vuelve a comprimirla y descomprimirla otra vez obtendrá una pérdida mayor. Cada vez que comprima y descomprima la imagen, ésta perderá algo de calidad. La compresión con pérdida no es conveniente en imágenes o gráficos que tengan textos o líneas y sobre todo para archivos que contengan grandes áreas de colores sólidos.

El algoritmo de compresión JPEG se basa en dos defectos visuales del ojo humano, uno es el hecho de que es mucho más sensible al cambio en la luminancia que en la crominancia, es decir, notamos más claramente los cambios de brillo que los de color. El otro es que notamos con más facilidad pequeños cambios de brillo en zonas homogéneas que en zonas donde la variación es grande, por ejemplo en los bordes de los cuerpos de los objetos.

6.1.2 Ruido producido por la compresión

El resultado tras la compresión, puede variar, en función de la agresividad de los divisores de la matriz de cuantización, a mayor valor de esos divisores, más coeficientes se convierten en ceros, y más se comprime la imagen. Pero mayores compresiones producen mayor ruido en la imagen, empeorando su calidad. Una imagen con una fuerte compresión (1%-15%) puede tener un tamaño de archivo mucho menor, pero tendrá tantas imperfecciones que no será interesante, una compresión muy baja (98%-100%) producirá una imagen de muy alta calidad, pero, tendrá un tamaño tan grande que quizás interese más un formato sin pérdida como PNG.

6.2 Imágenes BMP

Windows bitmap (.BMP) es el formato propio del programa Microsoft Paint, que viene con el sistema operativo Windows. Puede guardar imágenes de 24 bits (16,7 millones de colores), 8 bits (256 colores) y menos. Puede darse a estos archivos una compresión sin pérdida de calidad: la compresión RLE (Run-length encoding).

Los archivos con extensión .BMP, en los sistemas operativos Windows, representan la sigla BitMaP (o también Bit Mapped Picture), o sea mapa de bits. Los archivos de mapas de bits se componen de direcciones asociadas a códigos de color, uno para cada cuadro en una matriz de pixeles tal como se esquematizaría un dibujo de "colorea los cuadros" para niños pequeños. Normalmente, se caracterizan por ser muy poco eficientes en su uso de espacio en disco, pero pueden mostrar un buen nivel de calidad. A diferencia de los gráficos vectoriales, al ser reescalados a un tamaño mayor, pierden calidad. Otra desventaja de los archivos BMP es que no son utilizables en páginas web debido a su gran tamaño en relación a su resolución.

Dependiendo de la profundidad de color que tenga la imagen cada pixel puede ocupar 1 o varios bytes. Generalmente se suelen transformar en otros formatos, como JPEG (fotografías), GIF o PNG (dibujos y esquemas), los cuales utilizan otros algoritmos para conseguir una mayor compresión (menor tamaño del archivo).

Los archivos comienzan (cabecera o header) con las letras 'BM' (0x42 0x4D), que lo identifica con el programa de visualización o edición. En la cabecera también se indica el tamaño de la imagen y con cuántos bytes se representa el color de cada pixel.

6.3 Imágenes PNG

PNG (Portable Network Graphics) es un formato gráfico basado en un algoritmo de compresión sin pérdida para bitmaps no sujeto a patentes. Este formato fue desarrollado en buena parte para solventar las deficiencias del formato GIF y permite almacenar imágenes con una mayor profundidad de contraste y otros importantes datos.

Las imágenes PNG usan la extensión .png y han obtenido un tipo MIME (image/png) aprobado el 14 de octubre de 1996.

El método de compresión utilizado por el PNG es conocido como deflación (en inglés "Deflate algorithm"). El algoritmo es un sistema de compresión de datos sin pérdidas que usa una combinación del algoritmo LZ77 y la codificación Huffman. Fue originalmente definido por Phil Katz para la versión 2 de su herramienta de archivado PKZIP, y fue más tarde especificado como RFC 1951.

El algoritmo deflación está libre de todo tipo de patentes subsistentes, y esto, antes de que expirara la patente de LZW (el cual es usado en el formato de archivo GIF), ha llevado a su popularización y su uso en archivos comprimidos bajo gzip y archivos de imagen PNG, además del formato de compresión ZIP para el cual fue diseñado originalmente por Katz.

7. Presentación del problema:

El Reconocimiento Óptico de Caracteres (OCR), así como el reconocimiento de texto, en general son aplicaciones dirigidas a la digitalización de textos. Identifican automáticamente símbolos o caracteres que pertenecen a un determinado alfabeto, a partir de una imagen para almacenarla en forma de datos con los que podremos interactuar mediante un programa de edición de texto o similar.

Las dificultades que podemos encontrar a la hora de reconocer un texto tipografiado y cito según Wikipedia, *no se pueden comparar con las que aparecen cuando queremos reconocer un texto manuscrito.*

El reconocimiento de un texto manuscrito supone un desafío. Aunque se compone básicamente de caracteres individuales, la mayoría de algoritmos OCR no consiguen buenos resultados, ya que la segmentación de texto continuo es un procedimiento complejo.

Si partiéramos de una imagen perfecta, es decir, una imagen con sólo dos niveles de gris, el reconocimiento de estos caracteres se realizaría básicamente comparándolos con unos patrones o plantillas que contuvieran todos los caracteres posibles. Ahora bien, las imágenes reales no son perfectas, por lo tanto el Reconocimiento Óptico de Caracteres debe superar varios problemas:

Primero, el dispositivo que obtiene la imagen puede introducir niveles de grises al fondo que no pertenecen a la imagen original.

Segundo, la resolución de estos dispositivos puede introducir ruido en la imagen, afectando los píxeles que han de ser procesados.

Tercero, la distancia que separa a unos caracteres de otros, al no ser siempre la misma, puede producir errores de reconocimiento.

Cuarto, la conexión de dos o más caracteres por píxeles comunes también puede producir errores.

Todos los algoritmos actualmente en uso de Reconocimiento Óptico de Caracteres, son empleados para un reconocimiento tipográfico. Este algoritmo está basado en cuatro etapas: Binarización, Fragmentación o segmentación de la imagen, Adelgazamiento de las componentes y Comparación con patrones.

Binarización

La mayor parte de algoritmos de OCR parten como base de una imagen binaria (dos colores), por lo tanto es conveniente convertir una imagen de escala de grises, o una de color, en una imagen en blanco y negro, de tal forma que se preserven las propiedades esenciales de la imagen. De ahí el empleo de las imágenes con formato PNG utilizadas en este estudio. Una de las formas de hacerlo, es mediante el histograma de la imagen donde se muestra el número de pixeles para cada nivel de grises que aparece a la imagen. Para binarizarla tenemos que escoger un umbral adecuado, a partir del cual todos los pixeles que no lo superen se convertirán en negro (0) y el resto en blanco (255).

Mediante este proceso obtenemos una imagen monocroma donde quedan claramente marcados los contornos de los caracteres y símbolos que contiene la imagen. A partir de aquí podemos aislar las partes de la imagen que contienen texto.

Fragmentación o segmentación de la imagen

Este es el proceso más costoso y necesario para el posterior reconocimiento de caracteres. La segmentación de una imagen implica la detección mediante procedimientos de etiquetado determinista o estocástico de los contornos o regiones de la imagen, basándose en la información de intensidad o información espacial.

Permite la descomposición de un texto en diferentes entidades lógicas, que han de ser suficientemente invariables, para ser independientes del escritor, y suficientemente significativas para su reconocimiento.

No existe un método genérico para llevar a cabo esta segmentación de la imagen que sea lo suficientemente eficaz para el análisis de un texto. Aunque, las técnicas más utilizadas son variaciones de los métodos basados en proyecciones lineales.

Una de las técnicas más clásicas y simples para imágenes de niveles de grises consiste en la determinación de los modos o agrupamientos (“clusters”) a partir del histograma, de tal forma que permitan una clasificación o umbralización de los pixeles en regiones homogéneas.

Adelgazamiento de las componentes

Una vez aisladas las componentes conexas de la imagen, se les tendrá que aplicar un proceso de adelgazamiento para cada una de ellas. Este procedimiento consiste en ir borrando sucesivamente los puntos de los contornos de cada componente de forma que se conserve su tipología.

La eliminación de los puntos ha de seguir un esquema de barridos sucesivos para que la imagen continúe teniendo las mismas proporciones que la original y así conseguir que no quede deforme.

Se tiene que hacer un barrido en paralelo, es decir, señalar los píxeles no deseados para eliminarlos todos a la vez. Este proceso se lleva a cabo para hacer posible la clasificación y reconocimiento, simplificando la forma de las componentes.

Comparación con patrones

En esta etapa se comparan los caracteres obtenidos anteriormente con unos teóricos (patrones) almacenados en una base de datos. El buen funcionamiento del OCR se basa en gran medida en una buena definición de esta etapa. Existen diferentes métodos para llevar a cabo la comparación. Uno de ellos es el Método de Proyección, en el cual se obtienen proyecciones verticales y horizontales del carácter por reconocer y se comparan con el alfabeto de caracteres posibles hasta encontrar la máxima coincidencia.

Existen otros métodos como por ejemplo: Métodos geométricos o estadísticos, Métodos estructurales, Métodos Neuro-miméticos, Métodos Markovianos o Métodos de Zadeh.

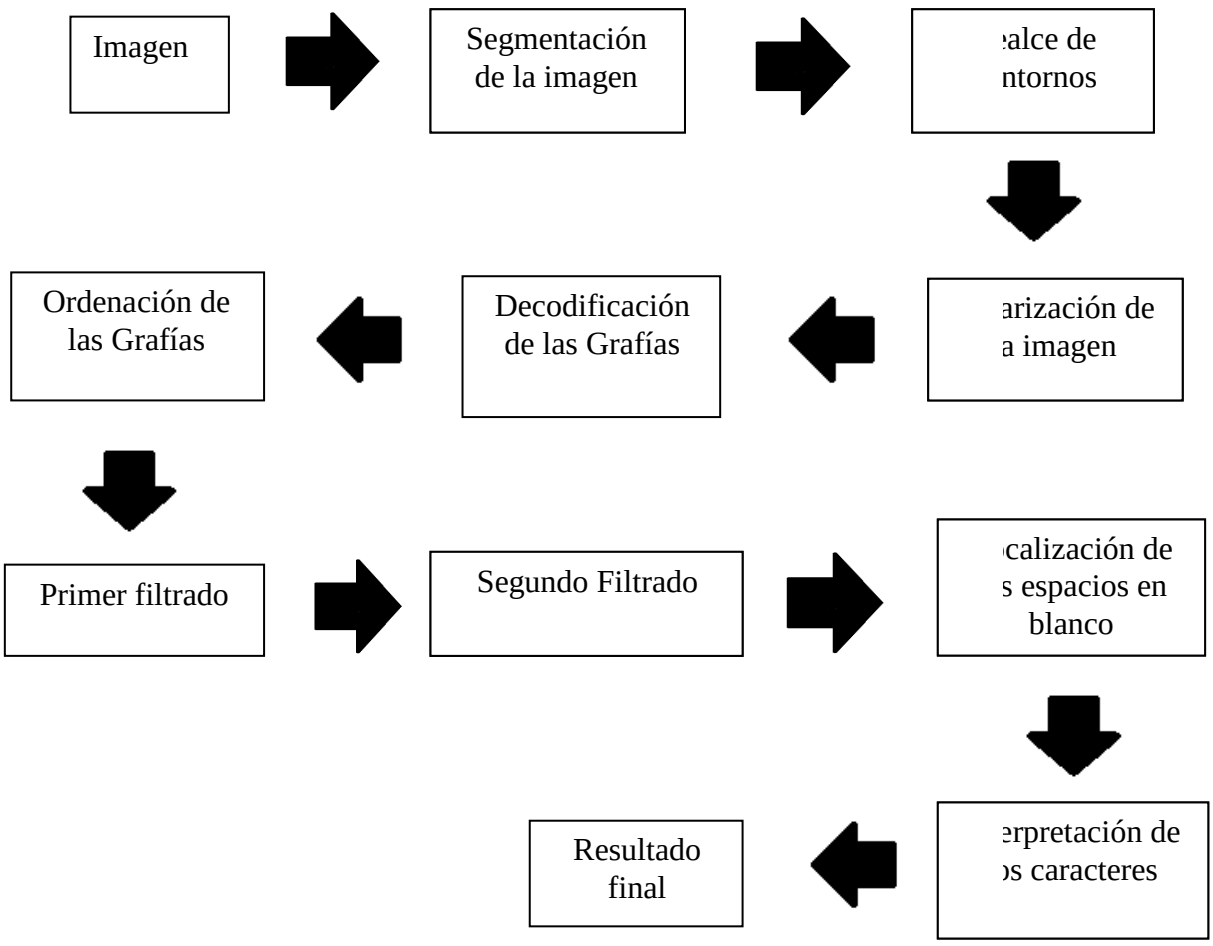
Aplicaciones

Desde la aparición de los algoritmos de Reconocimiento Óptico de Caracteres han sido muchos los servicios que han introducido estos procesos para aumentar su rendimiento y otros que se basan completamente en estas tecnologías. Algunos de los usos más comunes son: Identificación de matrículas, indexación de bases de datos y reconocimiento estructurado con OCR zonal.

8. Métodos Y Algoritmos

8.1 Programa Decodificador

8.1.1 Diagrama de Flujo



8.1.2 Segmentación de la hoja del códice:

Debido a que las imágenes presentadas para su decodificación muestran dos caras del códice, ha sido necesaria una segmentación previa de las mismas por la gran cantidad de grafías que poseen. Además se han eliminado los bordes de las hojas, ya que contienen información no relevante, que además ralentiza los algoritmos de búsqueda.

Por tanto en primer lugar se ha empleado la función **cvSetImageROI(imagen fuente, cvRect(x, y, ancho,alto))** . Como ya se ha indicado, esta función selecciona una región de la imagen fuente, en nuestro caso la hoja inicial del códice, según los parámetros definidos en el código. A continuación la guardará en la carpeta correspondiente del proyecto contenida a su vez el la carpeta de Visual Studio 2008. Esta función se implementa dos veces, una por cada cara de la hoja. Después son guardadas como "img_pagina_A.jpg" e "img_pagina_B.jpg" por la función **cvSaveImage(nombre del archivo, nombre de la imagen)**.

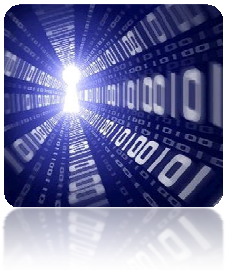
8.1.3 Binarización de la imagen:

Una vez hecho esto, se procede a la binarización de las imágenes. El motivo es que a pesar de la alta calidad de las mismas, las grafías no contrastan lo suficiente con el fondo de la hoja, y por tanto no pueden ser reconocidas por el programa decodificador. Para solventar dicho problema, se binariza la imagen según un cierto umbral óptimo encontrado: "umbral". De esta manera, aunque la imagen sigue conteniendo píxeles RGB, se ha potenciado la definición del contorno de las letras, al tiempo que se ha difuminado el fondo, y sus correspondientes texturas. Sin embargo, una de las formas de optimizar el tiempo de búsqueda del programa, reside en trabajar con imágenes en formato .BMP, es decir, monocromáticas. Por este motivo, se ha convertido la imagen binarizada a escala de grises, según el umbral "umbral2", para nuevamente volver a ser binarizada y convertirla al formato BMP. La imagen obtenida presenta una gran definición de sus grafías al tiempo que elimina prácticamente la totalidad del ruido presente en la imagen inicial.

Código:

```
cvThreshold(hoja_A, binarizada,umbral,v_max,CV_THRESH_BINARY); //Primera binarizacion
img_intermedia = cvCloneImage( binarizada );
cvCvtColor(img_intermedia, grises,CV_RGB2GRAY); // Conversión a escala de grises
img_final = cvCloneImage( grises );
cvThreshold(grises, img_final,umbral2,v_max,CV_THRESH_BINARY); //Segunda bianrización
cvSaveImage("img_lista.bmp",img_final);
```

8.1.4 Reconocimiento de las grafías:



El principal problema que se presentaba para la realización de este algoritmo, era la forma de analizar un conjunto variable de píxeles, de manera que la unión de los mismos conformara una grafía, una separación o un borde de página. Hemos de recordar que un microprocesador entiende una imagen como un array de longitud variable de 0 y 1, y no como la imagen que se nos muestra. Por este motivo, se decidió que la forma más sencilla de solucionar este problema, era comparar una matriz bidimensional de píxeles que conformara una única grafía, con un paquete de píxeles de igual tamaño de la imagen, es decir, dos cuadrículas o plantillas idénticas de tamaño entre si. De esta manera, era necesario crear un conjunto de plantillas, lo más extenso posible, que sirviera para identificar el mayor número de grafías de la imagen. Cuanto mayor sea este número, mayores posibilidades habrá de decodificar la totalidad de la imagen. No obstante, cabe destacar, que estas plantillas han de estar bien diferenciadas entre si, ya que dos plantillas altamente similares, originarán dos decodificaciones positivas para la misma grafía del texto, y en consecuencia mayor carga de trabajo para el resto de los algoritmos. Por tanto una plantilla o matriz bidimensional es un archivo, en este caso del tipo PNG, que cumple con las necesidades del programador y mediante el cual se realiza una identificación parcial 4646por semejanza con la imagen principal.

Mediante la función **cvMatchTemplate** de OpenCv es posible comparar matrices bidimensionales, es decir, plantillas definidas por el programador, con la imagen a analizar. Debido al elevado número de iteraciones que debe realizar el algoritmo, se decidió almacenar las plantillas como imágenes, en un vector de estructuras con un único campo del tipo IplImage*, óptimo para la definición de una imagen. Éste es a su vez una estructura con los siguientes campos, que definen las características más importantes de la imagen almacenada.

```
typedef struct _IplImage {
int nSize;   /* tamaño de la estructura iplImage */
int ID;    /* versión de la cabecera de la imagen */
int nChannels;
int alphaChannel;
int depth;    /* profundadad de la imagen en píxeles */
char colorModel[4];
char channelSeq[4];
int dataOrder;
int origin;
int align;   /* alineamiento de 4 u 8bytes*/
int width;
int height;
struct _IplROI *roi;   /* puntero a la ROI si existe */
struct _IplImage *maskROI;   /*puntero a la máxcara ROI si existe */
void *imageId;  /* uso de la aplicaión */
struct _IplTileInfo *tileInfo;   /* contains information on tiling
int imageSize;   /* tamaño útil en bytes */
char *imageData;   /* puntero a la imagen alineada */
int widthStep;   /* tamaño de alineamiento de linea en bytes */
int BorderMode[4];
int BorderConst[4]; /* constantes para el top, bottom, left, y right border */
char *imageDataOrigin; /* puntero a la imagen completa, sin alinear
```

*/
} IplImage;

A continuación se presenta el algoritmo completo de búsqueda e identificación de las grafías del texto.

```
for (c=0; c<num_letras;c++)  // "c" es el parámetro que identifica la plantilla que se está utilizando

{
/*Se obtiene el alto y el ancho para la imagen donde se almacenan los resultados  "res" */


 w = base_img[0].img->width  - base_img[1+c].img->width  + 1; //base_img[0] almacena la
 h = base_img[0].img->height - base_img[1+c].img->height + 1;// hoja del códice a analizar

/* Se crea la imagen para almacenar los resultados de la comparación */
res = NULL;
res = cvCreateImage( cvSize( w, h ), IPL_DEPTH_32F, 1 );

/*Se implementa la comparación de plantilla/imagen  */
cvMatchTemplate( base_img[0].img, base_img[1+c].img, res,  CV_TM_SQDIFF_NORMED);
/* Bucle de comparación */
for( y = 0 ; y < h ; y++ ) {
  for( x = 0 ; x < w ; x++ ) {
    /* obtención de un elemento */
    s = cvGet2D( res, y, x );

          if((y>1)&& (x>1)){
                anteriory=cvGet2D( res, y-1, x );
                anteriorx=cvGet2D( res, y, x-1 );}


          else {anteriory=cvGet2D( res, y, x );
          anteriorx=cvGet2D( res, y, x );
          }

    /*Si el valor está por debajo de "threshold",, hay una coincidencia con la plantilla */

                if( s.val[0] <= threshold ) {
                      if(( anteriory.val[0] <= threshold )|| (anteriorx.val[0]<=threshold)) {
                            x++};

                                                  else{
                                                        printf("no hay una imagen antes, dibujo rectangulo\n");

                                                        cvRectangle( base_img[0].img,
                                                  cvPoint( x, y),
                                                  cvPoint( x + base_img[1+c].img->width, y +
                                                  base_img[1+c].img->height ),
```

```
                                        cvScalar( 100,100,100,100),1 , 0, 0 );

                                        printf("intensity=%f\n",s.val[0]);

                                        texto[contador].posx = x;
                                                texto[contador].posy = y;
                                                texto[contador].l = arial[c];
                                    contador=contador+1;//Incrementamos el contador de
                                                }                       // letras encontradas
                        }
  }
}
}
```

8.1.4.1 Análisis del algoritmo:

Las funciones principales que conforman este algoritmo son:

cvMatchTemplate(Imagen a analizar, plantilla, imagen resultado, Método empleado)

Esta función, compara la imagen a analizar con al n-plantillas, y guarda el resultado en la imagen "res". Puede emplear distintos métodos de comparación.

Posibles métodos:

CV_TM_SQDIFF
CV_TM_SQDIFF_NORMED
CV_TM_CCORR
CV_TM_CCORR_NORMED
CV_TM_CCOEFF

CV_TM_CCOEFF_NORMED

Después de hacer pruebas con todos los métodos aquí expuestos, por la calidad de los resultados se eligió CV_TM_SQDIFF_NORMED. Detalle del algoritmo:

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

cvGet2D(res, y, x)

La función cvGet2D selecciona el píxel x,y de comienzo de cuadrícula a analizar de la imagen en la que se ha guardado el resultado de la comparación, "res" y le asigna un valor numérico que representa la similitud con la plantilla. Cuanto más pequeño sea dicho valor, más parecida es esa cuadrícula a la plantilla. El valor máximo a partir del cual dos cuadrículas se consideran distintas, queda fijado por la constante "threshold", en este caso 0.191, que tras

numerosos ensayos ha resultado ser el umbral óptimo. Cabe destacar que aunque hay proporcionalidad directa entre el valor dado al umbral “threshold” y el número de coincidencias, la fiabilidad es menor.

La función cvGet2D (res, y, x), devuelve enteros de tipo CvScalar*. CvScalar* es una estructura de enteros de la forma:
CvScalar* nombre de la variable {
Int val[0];
Int val[1];
Int val[2]};

Si se trata de una imagen monocroma, como es el caso de las imágenes analizadas, tan solo nombre de la variable .val[0] contendrá el valor devuelto por cvGet2D.

8.1.4.2 Estructuras empleadas

Struct letra *

Struct letra * será una estructura mixta de enteros y caracteres que contendrá la trascripción de las grafías encontradas, así como su ubicación cartesiana en la imagen a estudiar. Las variables que se definan como pertenecientes a esta estructura serán arrays de n posiciones.

Struct letra *{
Int posx;
Int posy;
Int ancho;
Char l};

Struct vec*

Struct vec* será la estructura utilizada para almacenar los archivos que recibe el programa. Está compuesta de un único campo de tipo char, es decir, caracteres.

struct vec*{
 char* arch;
 };

Struct imágenes*

Esta estructura sirve para definir variables o arrays con un único campo de tipo imagen, con a su vez, sus correspondientes subcampos. Recordemos que las variables definidas como IplImage* son a su vez estructuras.
En el programa se ha empleado para almacenar los archivos de imágenes con los que opera.

```
struct imagenes{
        IplImage* img;
};
```

8.1.4.3 Desarrollo del algoritmo:

Esté algoritmo, como todos los subsiguientes, se concibió en un principio, para una única plantilla, y por tanto para una única iteración. De esta forma la localización y eliminación de errores, que no suelen ser escasos, resultó mucho más sencilla. Una vez verificada la funcionalidad del algoritmo, se procedió a implementarlo para n-plantillas.

En primer lugar, el programa entra en un bucle, controlado por la variable “c” la cual se identifica numéricamente con la posición que ocupa cada plantilla en el vector de estructuras de imagenes base_img[i]. De forma que cuando c=0, la imagen guardada en base_img[] es la imagen a analizar, cuando c=1, la imagen de base_img[] es la primera plantilla de la “a” y así sucesivamente.

El siguiente paso almacena los valores de ancho “w” y alto “h” que deberá tener la imagen en la que se almacenará el resultado de la comparación, “res”. Es un método fijo, implantado por Intel que no admite ninguna posibilidad de cambio. El ancho de la imagen “res” será el ancho de la imagen a analizar menos el de la plantilla más 1. El procedimiento es análogo para el alto, h. Una vez se tienen estos dos parámetros, se crea la imagen “res” y se llama a la función cvMatchTemplate, la cual guardará la imagen fruto de la comparación en “res”. Como se ha podido ver en la explicación de esta función, existen diferentes métodosActo seguido, se entra en un doble bucle, controlado por la variables “x” e ”y” , utilizadas para desplazarse por las filas y columnas de píxeles de la imagen contenida en “res”. Si concebimos las imágenes como gran array bidimensional de “y” filas por “x” columnas, en lugar de como un vector unidimensional, resulta mucho más fácil situar espacialmente las acciones del algoritmo.

A continuación se llama a la función cvGet2D. Se omite su funcionamiento exacto por haberse explicado anteriormente. “Anteriorx” y “anteriory” , ambas variables del tipo CvScalar, almacenan el valor de similitud de la comparición entre la imagen analizada y la plantilla, un píxel por encima y un píxel antes de la cuadrícula con origen x,y. Ver detalle del código.

```
s = cvGet2D( res, y, x );

        if((y>1)&& (x>1)){
                anteriory=cvGet2D( res, y-1, x );
                anteriorx=cvGet2D( res, y, x-1 );}


        else {anteriory=cvGet2D( res, y, x );
        anteriorx=cvGet2D( res, y, x );
        }
```

De esta forma:

```
if( s.val[0] <= threshold ) {//Primera condicion
if(( anteriory.val[0] <= threshold )|| (anteriorx.val[0]<=threshold)) {  \\ Segunda condición
X++;
//No se encuentra coincidencia}
Else{….}
```

Si la comparación de cuadrículas supera el valor umbral (primera condición) y no hay otra coincidencia un píxel más a la izquierda o arriba (segunda condición), se encuentra una coincidencia.

Esta segunda condición resulta imprescindible, ya que si no, el programa encuentra numerosas coincidencias para la misma plantilla sobre idéntica grafía, que distan unas de otras tan solo un píxel vertical u horizontal. Mediante este condicionante, esa posibilidad indeseada, queda anulada.

Si el programa supera estas restricciones, es que hay una coincidencia positiva. Se ha decidido que para que puedan observarse las decodificaciones directamente sobre la imagen estudiada, el programa deberá realizar un rectángulo negro sobre las letras para las que ha encontrado una equivalencia. Al mismo tiempo, el algoritmo realiza la trascripción de las grafías encontradas, situándolas espacialmente, almacenando su anchura y asignándoles su correspondiente grafía en letra de imprenta. La variable "contador" almacena el número de coincidencias encontradas, para tener una noción de la longitud del vector de estructuras "texto".
Así mismo, será empleada en algunos de los subsiguientes algoritmos.
Ver detalle del código.

```
texto[contador].posx = x;
 texto[contador].posy = y;
texto[contador].ancho=base_img[c+1].img->width;
 texto[contador].l = arial[c];
contador= contador +1;  //Fin del algoritmo
```

El motivo por el que se ha diseñado el algoritmo de esta manera tan laboriosa, es porque ha resultado ser, después de numerosos ensayos, la forma más rápida de identificación y trascripción de grafías.

El programa localiza primero todas las grafías tipo "a" luego las tipo "b", tipo "c" y así sucesivamente. Para obviar la pérdida de la situación espacial en la imagen, se asigna a cada coincidencia unas coordenadas cartesianas x,y, con las que se ordenan las grafías posteriormente. En un principio, se pensó como alternativa en identificar en primer lugar la primera grafía de la primera línea de la imagen y de ahí en adelante. Sin embargo, esto comporta un tiempo de ejecución superior, ya que habría que posicionarse en el primer píxel de la imagen, y a partir de ahí comprobar con cada plantilla, si hay una coincidencia. Teniendo en cuenta que las longitudes de las plantillas varían entre sí, la situación se dificulta. Si la cantidad de plantillas fuese al menos 100, el número de iteraciones para cada nueva ubicación de la cuadrícula podría ascender a dicha cantidad, y por tanto, la cuantía de iteraciones totales sería muy elevada. Por este motivo se descartó esta alternativa.

8.1.5 Ordenación del vector de estructuras:

Como se indicó anteriormente, aunque el array "texto" contiene la decodificación ortográfica de la imagen, ésta se encuentra desordenada. Por ello ha sido necesario el diseño de un nuevo algoritmo que lo ordene a partir de las variables "posx" y "posy", que identifican la situación espacial de cada grafía en la imagen. Éste ha sido con diferencia el algoritmo más difícil de desarrollar de todo el proyecto. El array "texto", no puede ordenarse, como se pensó en un principio, en orden creciente según sus valores "x e "y". El hecho de que la variable "y" de una letra posea mayor valor que la de otra, por estar situada más abajo, no implica que deba tener prioridad en la ordenación. Éste es el caso de una "a" seguida en la imagen de una "h" (a h) que en el vector "texto" aparece como "h" seguida de "a" (h a). Como "h" posee mayor altura, su "y" es menor, puesto que está situada más cerca del borde superior. Por tanto, si siguiéramos un criterio de ordenación creciente, "h" debería ocupar una posición inferior en el array texto, y por tanto, en el array "texto" estarían correctamente ordenadas. Sin embargo, como puede observarse, esto no es así. Cuando nos encontramos en esta coyuntura, para solucionar esta excepción, se ha optado por introducir una variable que mide la diferencia de alturas entre dos letras. Si la diferencia entre "h" y "a" u otras dos letras cualesquiera, es menor que un determinado factor, consideramos que se encuentran en la misma línea y por tanto tendrá prioridad aquella con menor valor de "x" a pesar de estar situada más arriba.

La otra excepción la encontramos para el caso contrario. Tanto en la imagen como en el vector "texto", aparece "h" seguida de "a". Sin embargo, en la imagen, "a" está muy por encima de "h" y por tanto "a" es prioritaria a pesar de tener mayor valor de "x". Esta característica debe ser respetada no obstante, si "h" y "a" se encuentran en la misma línea. Por tanto, para poder tener un criterio fiable, se ha introducido nuevamente una variable que calcule la diferencia de alturas, y en caso de ser superior, al contrario que antes, a un determinado factor, otorgarle prioridad en la ordenación a la segunda letra.

Detalle del código:

```
//Función de ordenación del array de estructuras texto

  int primero, segundo;

for (primero=0; primero<contador-1;primero++)
 {
   for (segundo=primero+1; segundo<contador; segundo++)
        {
        if(texto[primero].posx > texto[segundo].posx  )
        {
                if (texto[segundo].posy <= texto[primero].posy){

                 aux=texto[primero];


                 texto[primero]= texto[segundo];
                 texto[segundo]=aux;
                                }
                if (texto[segundo].posy > texto[primero].posy){    //excepciones
                        diff=texto[segundo].posy - texto[primero].posy;
                        if(diff<=factor){
                                aux=texto[primero];
                                texto[primero]= texto[segundo];
                                texto[segundo]=aux;
                                        }
                }
        }
        else if (texto[primero].posx < texto[segundo].posx){

                if (texto[segundo].posy < texto[primero].posy){
                        diff= texto[primero].posy - texto[segundo].posy;   //excepciones
                        if (diff>factor){

                                aux=texto[primero];
                                texto[primero]= texto[segundo];
                                texto[segundo]=aux;
                                                }
                }
        }
        else {
                if (texto[segundo].posy< texto[primero].posy){

                aux=texto[primero];
                texto[primero]= texto[segundo];
                texto[segundo]=aux;
                                }
        }
```

```
        }
}
```

Como puede observarse, no ha resultado fácil encontrar un criterio que permita una ordenación sistemática del array. Los casos que no aparecen contemplados no implican un cambio en su ordenación.

8.1.6 Filtrado del array de estructuras texto

8.1.6.1 Primer filtrado:

Una vez se ha obtenido la decodificación ortográfica de la imagen, y se ha ordenado, es necesario filtrar las coincidencias encontradas. Según se ha observado, en algunas grafías se encuentran varias coincidencias para distintas plantillas. Esto se debe a que en ocasiones, la diferencia existente entre dos plantillas es insuficiente como para ser consideradas dos grafías diferentes, por lo que el programa encuentra dos o más coincidencias para una misma grafía de la imagen. Por esta razón, se ha ideado un algoritmo que, mediante un primer filtrado, localice aquellas letras que estén repetidas y les asigne un carácter especial. En este caso "#". La forma de identificar estos errores de trascripción, ha sido mediante el cálculo de las distancias tanto en "x" como en "y", entre dos letras consecutivas del array "texto". Si la diferencia es menor que un cierto valor, "factor_2", y su correspondiente trascripción a letra de imprenta coincide, se considera que es la misma letra pero con una desviación espacial de uno o dos píxeles, y por tanto se le asigna a la primera letra el carácter especial "#". No obstante, aunque resulta un algoritmo muy fiable, hay que iterarlo suficientes veces como para que pueda eliminar todos los errores. Se ha decidido que con diez iteraciones es suficiente, ya que pueden eliminarse hasta 19 repeticiones consecutivas de una misma letra.

Otro aspecto a tener en cuenta son las consonantes dobles. En caso de que en la imagen apareciese una doble "l" o "r", no debería considerarse como error, o al menos no siempre. Por ello, se ha incluido una condición especial para estas situaciones. De manera que solo se filtrarán las consonantes dobles, "l" o "r", cuando aparezcan al menos tres de las mismas, de forma consecutiva. De ser así, procede a eliminarse la primera, como sucedía en el caso general.

```
int dif_x=0;
int dif_y=0;
        for (c=0;c<10;c++){
                for(i=0;i<contador;i++){
                        if(texto[i].l== texto[i+1].l){
                                dif_x= texto[i].posx- texto[i+1].posx;
                                dif_y=texto[i].posy- texto[i+1].posy;

                                if (dif_x<0){dif_x=dif_x * (-1);}//Se calcula en valor absoluto
                                if (dif_y<0){dif_x=dif_y * (-1);}
```

```
                        if((dif_x < factor_2) || (dif_y<factor_2)){ //menor de dos pixeles
                                //Hay al menos una letra consecutiva repetida
                                if((texto[i].l !='l' && texto[i+1].l!='l')
                                        ||(texto[i].l != 'r' && texto[i+1].l !='r')){
                                                texto[i].l=R; // Carácter especial
                                repetida= repetida +1;
                                }


                        else if((texto[i].l =='l' && texto[i+1].l=='l' && texto[i+2].l=='l')
                        ||(texto[i].l == 'r' && texto[i+1].l =='r' && texto[i+2].l =='r')){
                                        //Si hay tres 'r' o 'l' consecutivas
                                                texto[i].l=R;
                                                repetida= repetida +1;
                                }
                        }
                }
        }
}
```

8.1.6.2 Segundo filtrado:

Este algoritmo se encargará únicamente de identificar el carácter especial “#”, eliminarlo del array “texto”, y reubicar el resto de las trascripciones para que no queden espacios vacíos.

```
int cuenta=0;

for (i=0;i<contador;i++){

        if(texto[i].l != R){

        texto_2[cuenta].l=texto[i].l;

        cuenta= cuenta +1;
}
}
```

8.1.7 Algoritmo de localización de espacios en blanco

Aunque en un primer momento este algoritmo fue diseñado de forma distinta a la presentada, los resultados obtenidos no fueron los esperados y por tanto tubo que descartarse e implementarse un método de resolución diferente. Más adelante se presentará el citado método fallido.

Se decidió que una posible solución al problema era que, una vez las grafías habían sido ordenadas y filtradas, un algoritmo calculase la distancia entre una letra y la siguiente. De manera que si esta superaba un cierto valor, el programa considerase que existía una separación entre ambas. La forma de calcular esta distancia consiste en restar a la coordenada de abcisas de la segunda letra, la suma del ancho de la primera letra y de su coordenada de abcisas. Recordemos que las variables que contienen esta información son "texo[ ].posx" y "texto[ ].ancho". Por tanto la operación sería:

```
espacio= texto_2[i+1].posx - (texto_2[i].posx  + texto_2[i].ancho);
```

Donde "espacio" almacena la diferencia, e "i" indica la posición de la celda en la que nos encontramos. Se observó que en ocasiones, esta diferencia era negativa, y se comprobó se debía a dos situaciones.

La primera era aquella en la que se restaban las coordenadas de dos letras que a pesar de estar ordenadas de forma consecutiva, la segunda, poseía una abcisa menor por encontrase en la siguiente línea.

La segunda sitación anómala se daba cuando el programa, a pesar de encontrar coincidencias para dos letras distintas, solapaba en parte el comienzo de la segunda letra con la de la primera, y por tanto, al realizar la operación de diferencia, el resultado era negativo aunque próximo al cero. La solución propuesta ha sido que en caso de que la diferencia supere los 20 píxeles o los -7, se considerará que existe un espacio entre las dos letras.

```
if((espacio >= factor_3)||(espacio<=-7)){ //Se considera que existe una separación
                    texto_3[i+1+inc].l= '  ';//Representación de espacio
                    texto_3[i+2+inc]=texto_2[i+1];
                    inc=inc+1;
    {…}
```

La variable "inc" contabiliza el número de espacios añadidos al array "texto_3[ ]". Se emplea para tener una noción de cuanto aumenta dicho vector y así ser más concretos en las iteraciones.

Es necesario resaltar que aunque la solución propuesta ha resultado ser válida en la mayoría de los casos, no es todo lo rigurosa que sería deseable.

8.1.8 Algoritmo de interpretación de los caracteres especiales:

Si se observa la imagen analizada, podrá comprobarse que existen algunas grafías que no tienen equivalente ortográfico en el alfabeto europeo. Esto se debe, como ya se explicó, a que dichas grafías representan ligaduras de dos o más letras. El problema de trascribir estos caracteres reside en el hecho de que el programa está diseñado para que cada plantilla represente una sola letra. Como ya es sabido, una vez encontrada una coincidencia el programa le asigna una letra tipo arial, contenida en el array con el mismo nombre. Si dependiendo del caso hubiera que asignar uno, dos o más caracteres a dicho vector, se producirían solapamientos, problemas de memoria y además, sería difícil de interpretar de forma rigurosa. Sin embargo, se encontró una solución relativamente sencilla a tan aparentemente complicado problema. Se decidió otorgar un carácter especial a estas ligaduras, de manera que a "et" se le ha asignado el carácter " * " , a "per-" el carácter "$" , a "pro" " - " y a "rum". " _" .

Una vez que el programa ha localizado los espacios en blanco, realiza un barrido del vector "texto_3[ ]" buscando los citados caracteres especiales. Copia cada carácter, espacios en blanco incluidos, a un vector análogo llamado "texto_4[ ]". Cuando encuentra un carácter especial, lo interpreta y lo trascribe a dicho array deshaciendo la ligadura. Ver detalle del código.

```
char et[]="et";
char per[]="per";
char pro[]="pro";
char rum[]="rum";
int cuenta2=0;

for(i=0;i<largo2;i++){

inc=0;
if((texto_3[i].l != '*')&&(texto_3[i].l != '$')&&(texto_3[i].l != '-')&&(texto_3[i].l != '_'))
                {

                texto_4[cuenta2].l=texto_3[i].l;

                cuenta2= cuenta2 +1;
                }

                else if(texto_3[i].l=='*'){//En caso de equivalencia, se deshace la

                        for(x=0;x<2;x++){//abreviatura. El numero de caracteres a copiar,

                                texto_4[cuenta2+inc].l=et[x];//dependerá del numero de
                                                        // caracteres  que haya que incluir
```

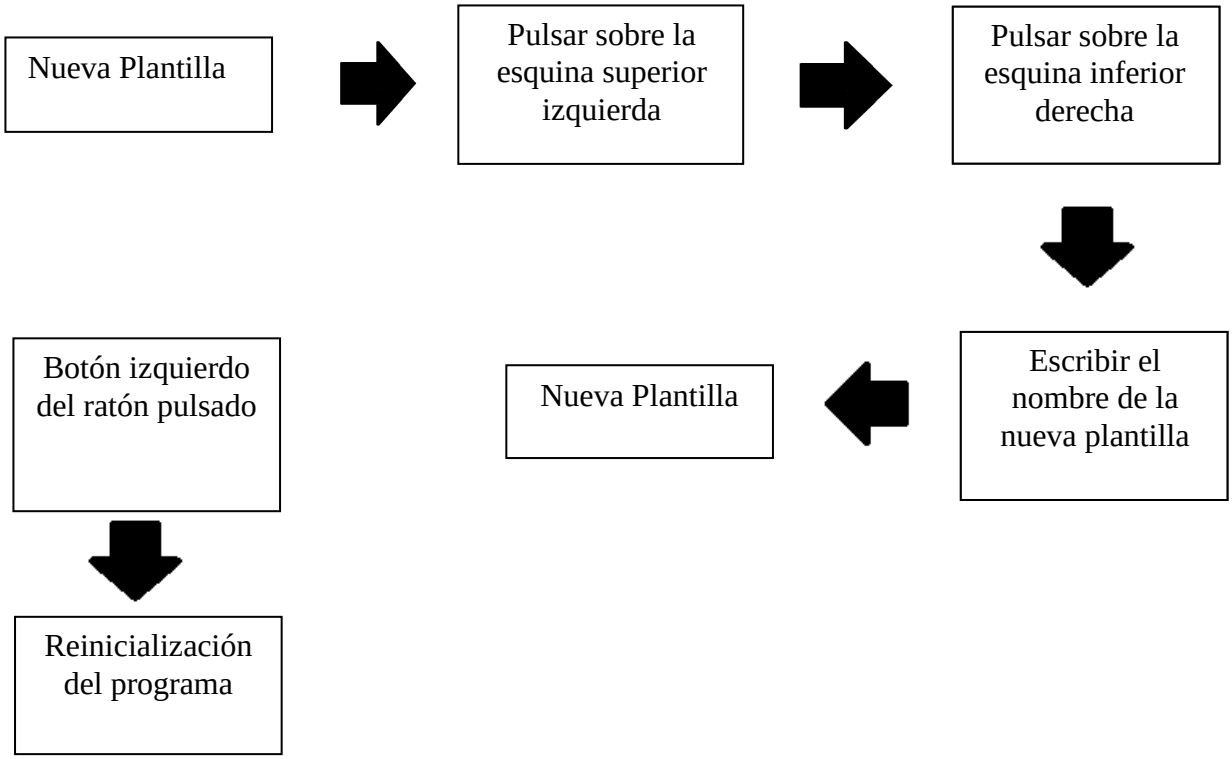
```
inc=inc+1;}
cuenta2=cuenta2+inc;

{…}
```

Esto se repite para cada carácter especial.

Una vez han finalizado los algoritmos de decodificación, se procede a guardar el resultado en el fichero"solución.txt" para una mayor comodidad de análisis y modificación. No obstante, durante todo el tiempo de ejecución pueden observarse en la consola los resultados de las operaciones que realiza el programa.

8.2 Programa de guardado automático de plantillas

8.2.1 Diagrama de Flujo



8.2.2 Desarrollo del algoritmo

A través del análisis de los datos obtenidos, los cuales serán presentados en el epígrafe de resultados de esta memoria, se comprobó que la calidad de la decodificación dependía no solo de los algoritmos y los valores umbrales introducidos, sino sobre todo, del número de plantillas inicial. El motivo es la gran diferencia que existe en la caligrafía de unos textos a otros e incluso, entre letras iguales. Por ello se decidió implementar un código que resultase de fácil familiarización para una persona ajena a la programación y que le permitiese aumentar el número de plantillas con las que cuenta el algoritmo.

Este programa comienza mostrando en una ventana la imagen de la que se desea obtener una o varias plantillas. Indica al usuario que proceda a pulsar sobre la esquina superior izquierda de la plantilla que desea guardar. A continuación, debe pulsar sobre la esquina inferior derecha. El programa mostrará por pantalla los valores numéricos de la plantilla y la imagen en sí. En caso de error al seleccionar la primera coordenada, el usuario puede resetear los valores pulsando el botón derecho del ratón. Así mismo, si altera el orden de los pasos, aparecerá un mensaje de error y el usuario deberá empezar de nuevo.

El siguiente paso será introducir el nombre con el que desea guardar la plantilla. Se escribirá únicamente el nombre, sin extensión. Tras pulsar enter, el programa confirmará el nombre con el que se ha guardado y volverá al estado inicial. Cuando desee finalizar, pulse cualquier tecla

El algoritmo está basado en la función **mouseHandler** de OpenCv. Esta función se emplea para detectar eventos, producidos bien por el ratón bien por el teclado. En este caso, se ha elegido como evento propiciatorio de la acción, la pulsación del botón izquierdo del ratón.

La segunda función empleada ha sido **cvSetImageROI(imagen fuente, cvRect(x, y, ancho,alto))**. Con esta función, al igual que en el algoritmo de segmentación de la imagen del programa principal, seccionamos la imagen definida por el usuario y la guardamos mediante la función **cvSaveImage(),** mostrándola en una ventana auxiliar. (4), (5), (6)

```
void mouseHandler(int event, int x, int y, int flags, void* param)

{


   /* Seleccion de la esquina superior izquierda de la plantilla */

   if ((event == CV_EVENT_LBUTTONDOWN) && (drag==0))
   {
                cvShowImage("img", img0);
                cvShowImage("img2", plantilla);
        point1 = cvPoint(x, y);
```

```
        drag  = 1;
                printf("X1= %i  Y1=%i, \n",point1.x, point1.y);
                printf("Ahora en la esquina inferior derecha de la imagen que desea guardar\n");
                printf("\n");

        }
                /* Seleccion de la esquina inferior izquierda de la plantilla */


        else if ((event == CV_EVENT_LBUTTONDOWN) && (drag==1))
                                        {

                cvShowImage("img", img0);
                        point2 = cvPoint(x, y);
                         drag  = 2;
                        printf("X2= %i  Y2=%i, \n",point2.x, point2.y);

        }

                                /*Creacion de la nueva plantilla*/

         else if ((event == CV_EVENT_LBUTTONUP) && (drag==2))
         {

                                x=point1.x;
                                y=point1.y;
                                w=point2.x -point1.x;
                                h=point2.y -point1.y;
                                if((w<=0)||(h<=0)){ printf("Valores mal introducidos, lea las instrucciones de nuevo por favor\n");
                                drag=0;}
                                else{
                                cvSetImageROI( img0, cvRect(x, y, w, h));

                                plantilla = cvCreateImage(cvGetSize(img0),img0->depth,img0->nChannels);
                                cvCopy(img0, plantilla, NULL);
                                cvResetImageROI(img0);
                                cvShowImage("img2", plantilla);
                                cvShowImage("img",img0);

                                printf("\n");
                                printf("Los valores de su nueva plantilla son:\n");
                                printf("\n");
                                printf("X= %i  Y= %i  W= %i  H= %i\n",x, y,w,h);
                                printf("\n");
                                cvShowImage("img2", plantilla);
                                cvShowImage("img",img0);
```

```
                                printf("Pulse sobre la imagen principal para continuar\n");
                                drag=3;}

        }
                                else if((event == CV_EVENT_LBUTTONDOWN) && (drag==3))
         {
                                cvShowImage("img2", plantilla);
                                cvShowImage("img",img0);
                                printf("Presione sobre la consola, introduzca el nombre de la
nueva planilla "tmp_letraNUMERO"\n");
                                printf( "y pulse enter\n");
                                scanf( "%s", nombre );
                                strcat (nombre,extension);
                                printf("Su plantilla se ha guardado como: %s\n",nombre);

                                cvSaveImage(nombre, plantilla);
                                printf("\n");
                                printf("Pulse nuevamente sobre la esquina superior derecha de la
plantilla que desee guardar\n");
                                drag = 0;
        }



         else if (event == CV_EVENT_RBUTTONDOWN)

   {
      cvShowImage("img", img0);
                  cvShowImage("img2",plantilla);
      drag = 0;
   }
```

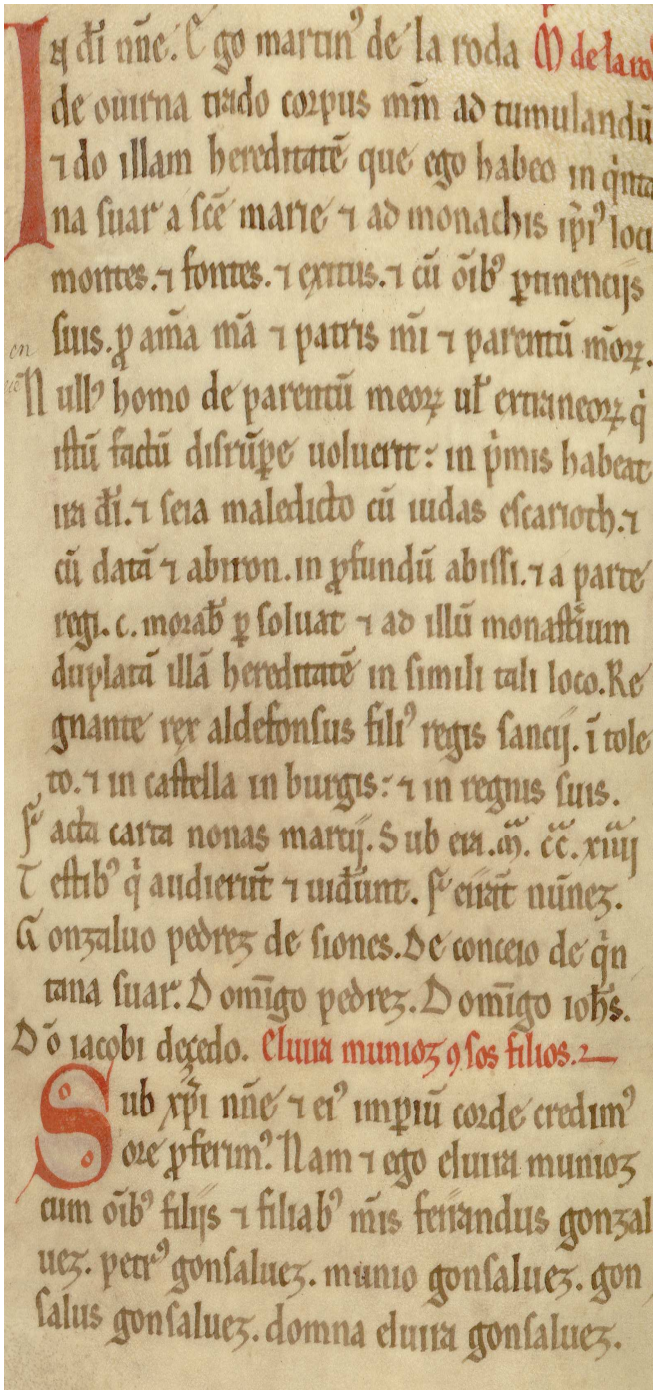
9. Resultados experimentales y discusión:

Los resultados son presentados según el orden de ejecución del programa.

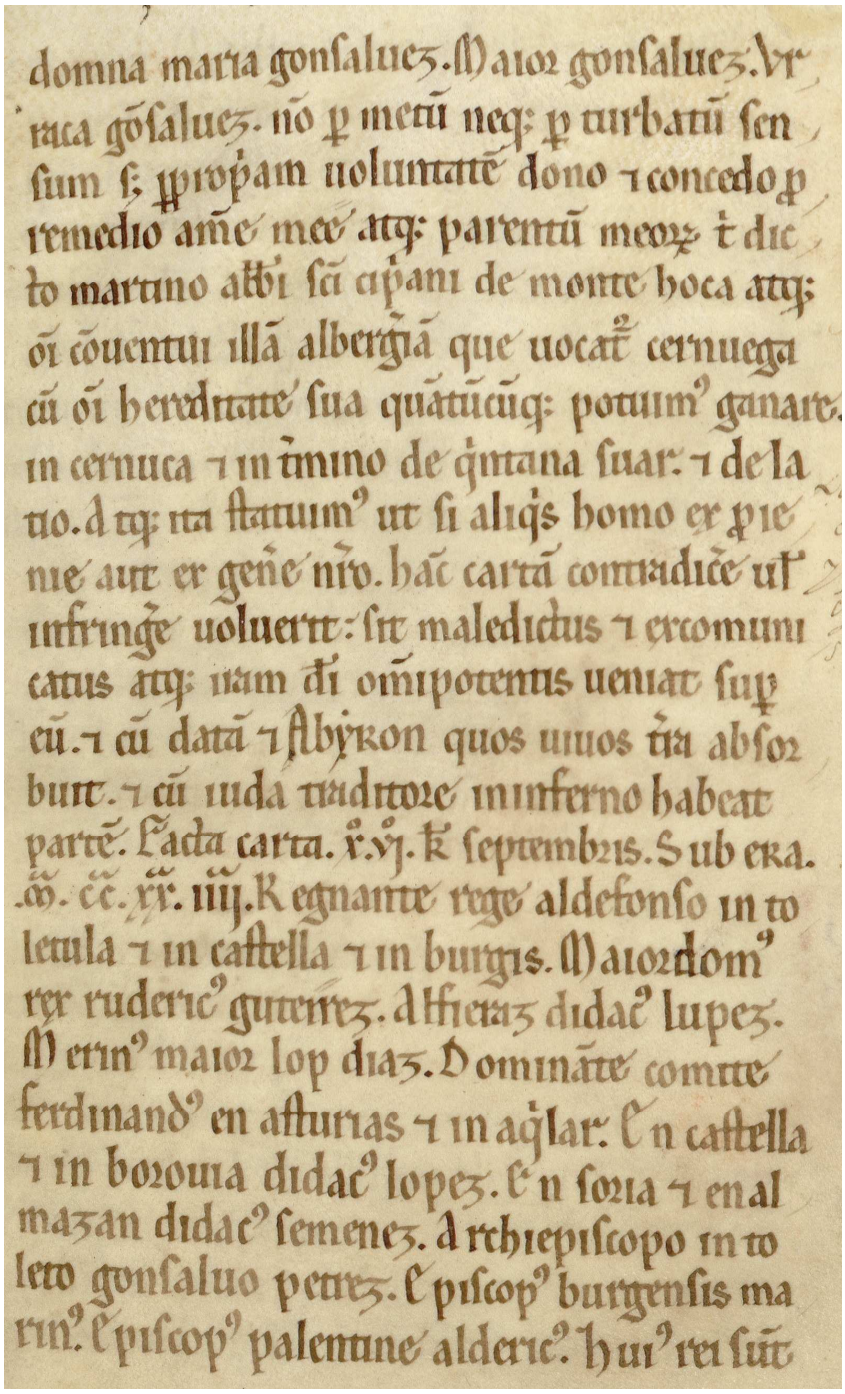
9.1 Segmentación de la imagen inicial

En este primer resultado, se puede observar la segmentación de la hoja Avr. en las dos caras que conforman la imagen. Se decidió realizar esta operación previa para eliminar los contornos de la imagen, con información prescindible, y así agilizar la labor de búsqueda del programa.

Cara izquierda:



Cara derecha:

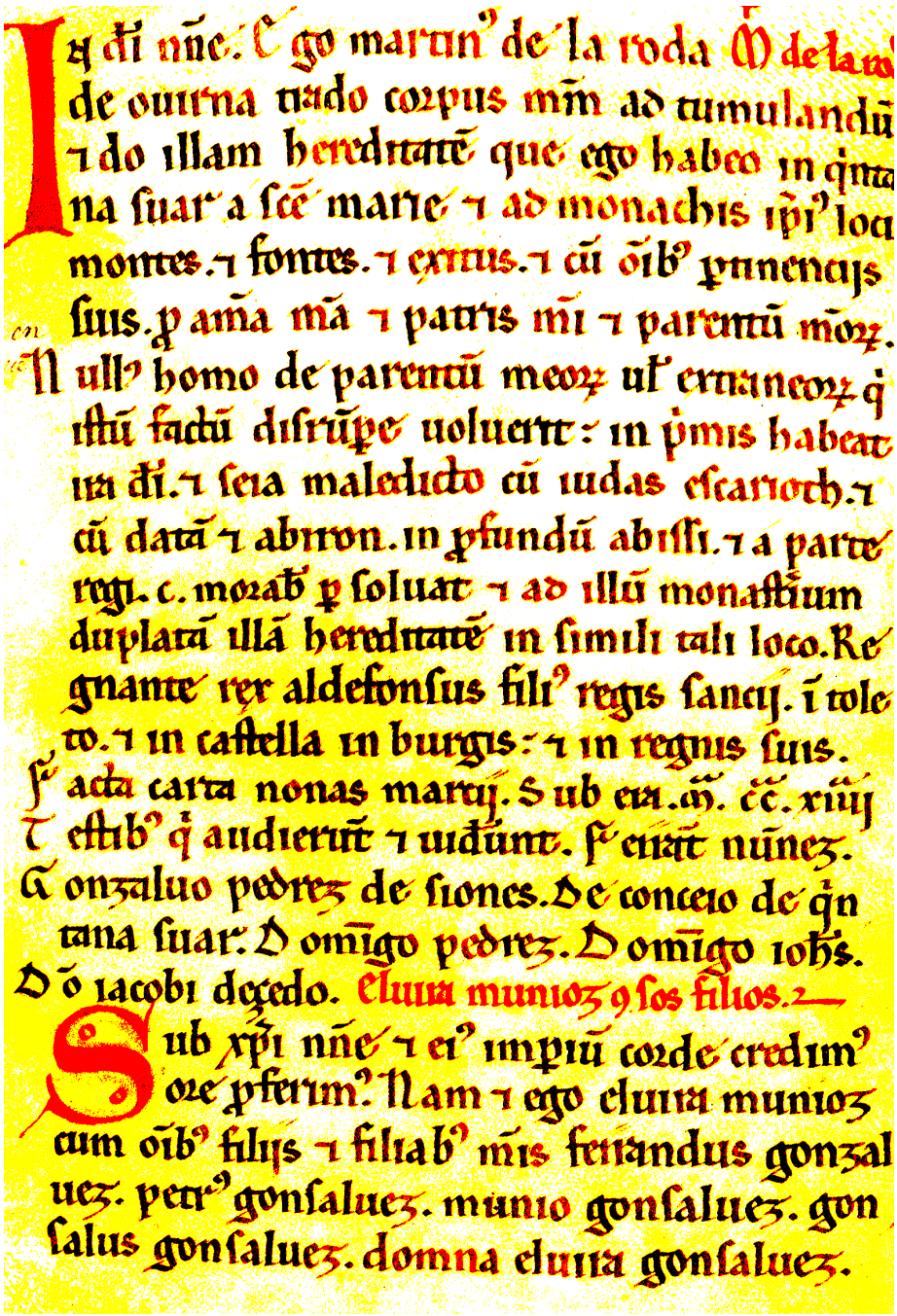


domna maria gonsaluez. ſ) maior gonsaluez. Vr
raca gõsaluez. nõ p̄ metũ neq; p̄ turbatũ sen
sum ſ; p̄ propiam uoluntatẽ dono ⁊ concedo p̄
remedio aīe mee atq; parentũ meorꝝ t dic
to martino abbi sci apostoli de monte hoca atq;
oi cõuentui illã albergiã que uocat cernuega
cũ oi hereditate sua quãtũcũq; potuim⁹ ganare.
in cernuca ⁊ in t̃mino de q̃ntana suar. ⁊ de la
tio. ā tq; ita statuim⁹ ut si aliqs homo ex p̄ie
nie aut ex genẽ nr̃o. hãc cartã contradicẽ uel
infringẽ uoluerit: sit maledictus ⁊ excomuni
catus atq; nam di omnipotentis ueniat sup̄
eũ. ⁊ cũ datã ⁊ abyron quos uiuos tra absor
buit. ⁊ cũ iuda traditore in inferno habeat
partẽ. ſacta carta. ř.vj. k̄ septembris. Sub era.
.cõ. cc. řř. iiij. Regnante rege aldefonso in to
letula ⁊ in castella ⁊ in burgis. ſ) maior dom⁹
rex ruderic⁹ gutierrez. archiepiscopus didac⁹ lupez.
ſ) erim⁹ maior lop diaz. Dominante comite
ferdinand⁹ en asturias ⁊ in aḡlar. ē n castella
⁊ in borouia didac⁹ lopez. ē n soria ⁊ en al
mazan didac⁹ semenez. archiepiscopo in to
leto gonsaluo petrez. ē piscop⁹ burgensis ma
rin⁹. ē piscop⁹ palentine alderic⁹. hui⁹ rei sũt

9.2 Primera binarización de la cara_A

Tras la segmentación se ha binarizado la cara A de la imagen. Puede observarse como se realzan los contornos de las letras al tiempo que las partes de la imagen que contienen información no relevante, como los espacios entre líneas, pierden definición.
Se comprobó que al aumentar el umbral, se potenciaba la definición de las grafías aún más. Sin embargo, también aparecían numerosos píxeles que deterioraban la calidad total de la imagen, no solo alrededor de las propias grafías, sino también entre la separación de las mismas. De igual manera, si el umbral resultaba ser demasiado bajo, había una pérdida completa de información en toda la imagen. Por este motivo se consideró que un umbral de 153, resultaba óptimo, ya que definía de forma clara las grafías, eliminaba en gran parte el fondo de hoja y a penas producía ruido.

Imagen óptima:



Detalle de la imagen:

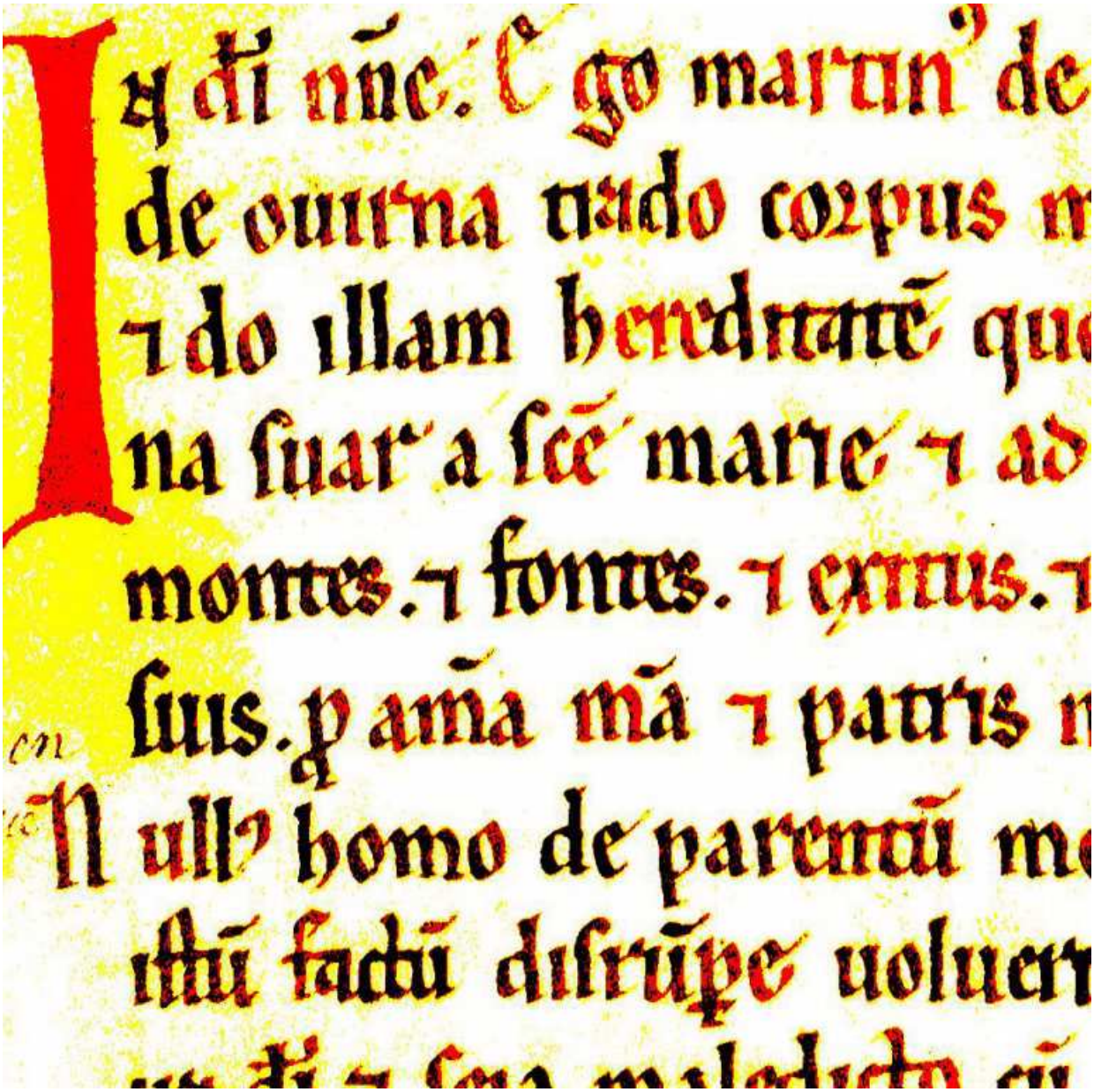


Imagen Sobreumbralizada:

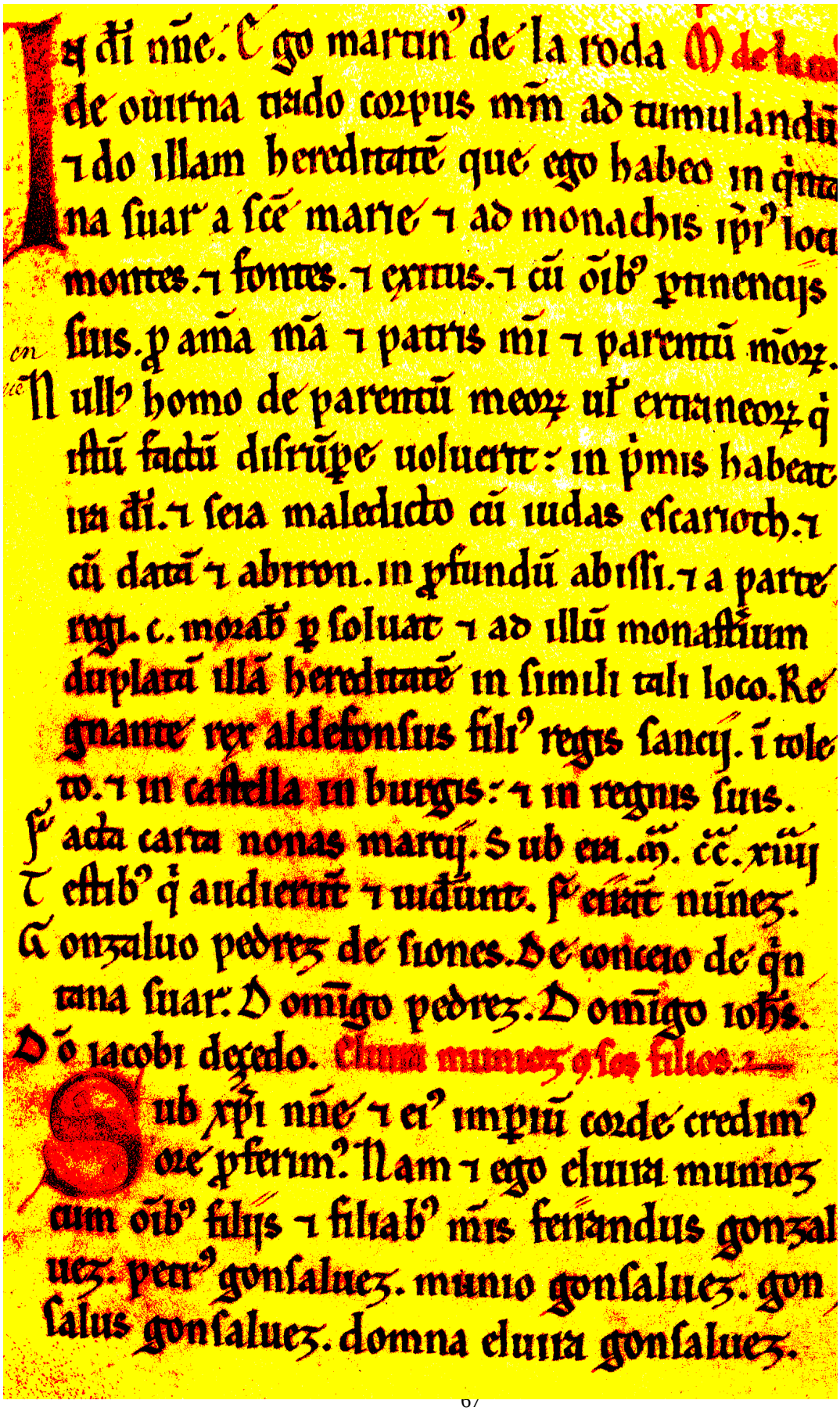
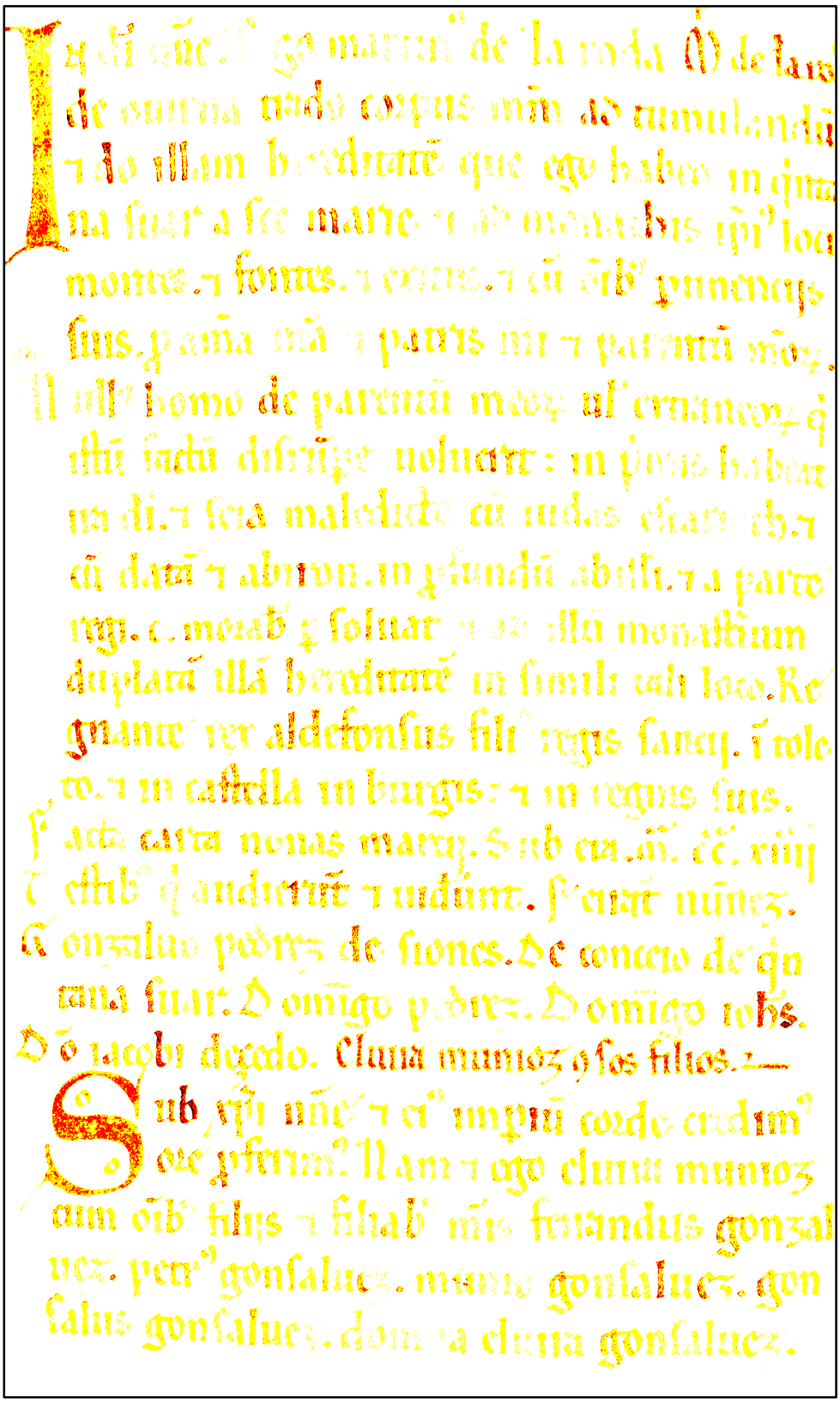

In dñi nñe. Ego martin' de la roda
de ouirna trado corpus mñm ad tumulandū
& do illam hereditatē que ego habeo in q̄nta
na suar a scē marie & ad monachis ip̄i' loc
montes. & fontes. & exitus. & cū oīb' pertinencijs
suis. p̄ aīa mā & patris mi & parentū mor.
Nullꝰ homo de parentū meor uꝫ extraneor q̄
istū factū disrūpe uoluerit: in p̄mis habeat
ira dñi. & sera maledicto cū iudas escarioth. &
cū datā & abiron. in p̄fundū abissi. & a parte
regi. c. morab' p̄ soluat & ad illū monastīum
duplatā illā hereditatē in simili tali loco. Re
gnante rex aldefonsus filiꝰ regis sancij. i tole
to. & in castella in burgis: & in regnis suis.
Facta carta nonas marcij. Sub era. m̄. cc. xliij
& ecclīb' q̄ audierūt & uiderūt. Fernā nūnez.
Gonzaluo pedrez de sones. De concejo de q̄n
tana suar. Domīgo pedrez. Domīgo iohs.
Dō iacobi decedo. Eluira munioz & los filios. &
Sub xp̄i nñe & ei' imp̄iū corde credim'
ore p̄fitim'. Nam & ego eluira munioz
cum oīb' filijs & filiab' mis fernandus gonzal
uez. petr' gonsaluez. munio gonsaluez. gon
salus gonsaluez. domna eluira gonsaluez.

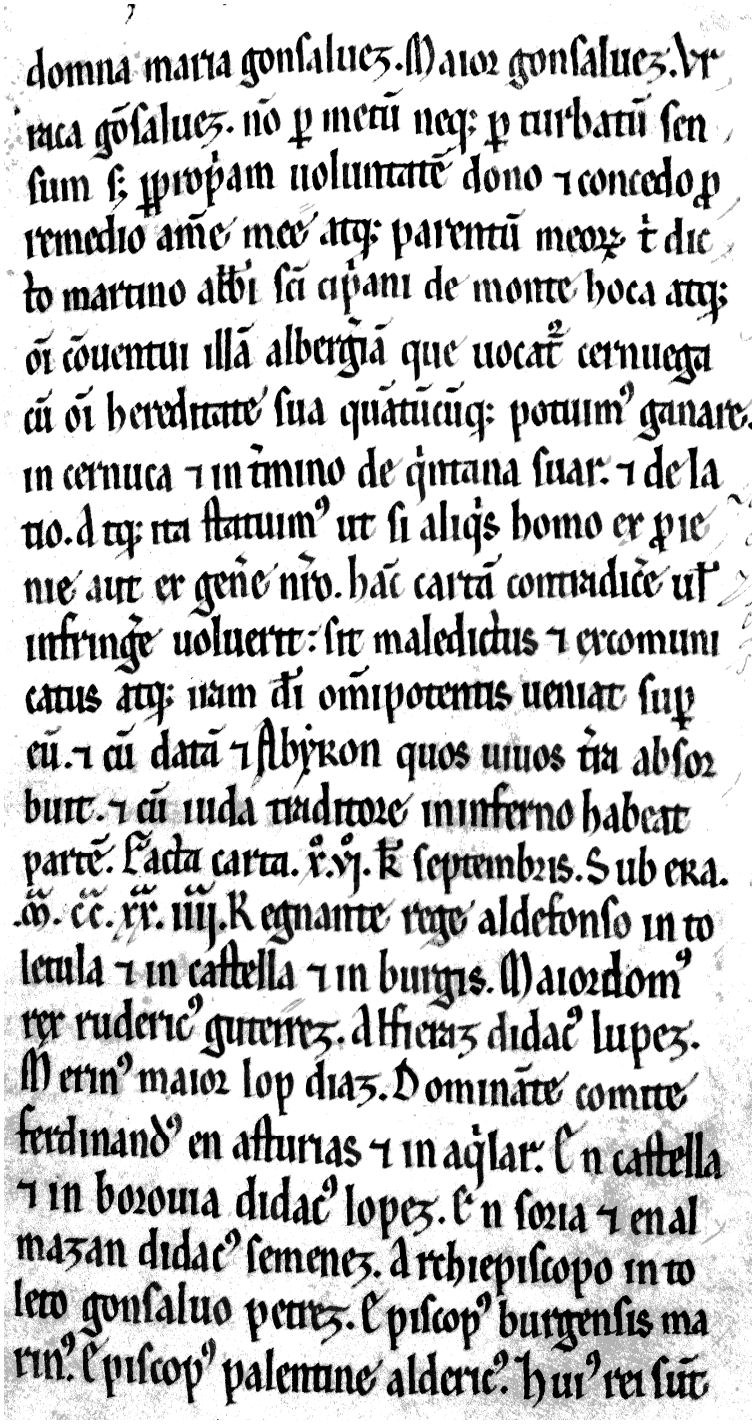
Imagen Subumbralizada:



9.3 Conversión a escala de grises

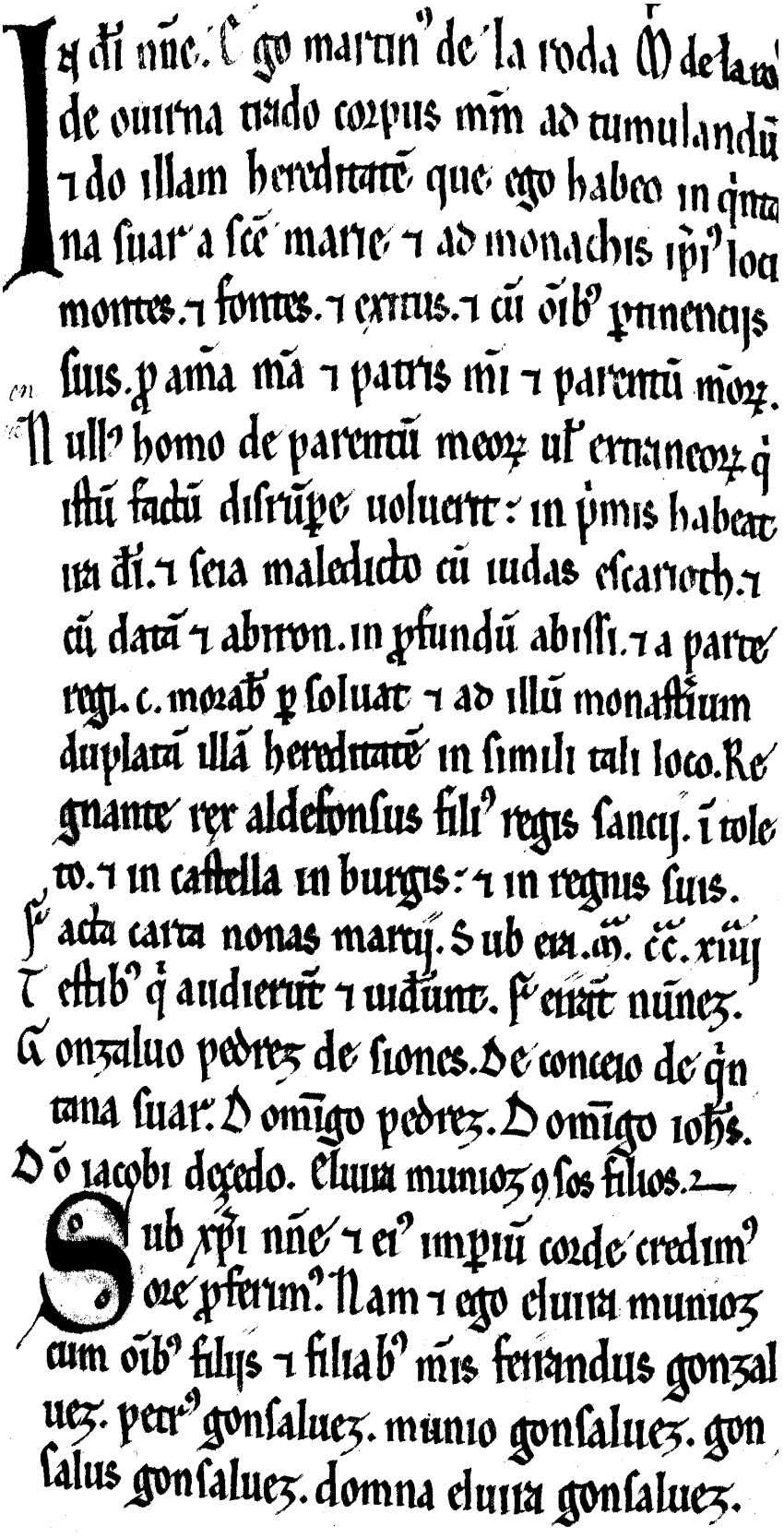
En el siguiente resultado puede observarse como se transforma la anterior imagen a escala de grises. Aquellos píxeles que estén por debajo del umbral, en este caso de 40, se considerarán negros, y los que lo superen, serán grises en sus distintas tonalidades. Recordemos que la finalidad de estas operaciones es trabajar la imagen con dos tipos de píxel 0 ó 1, de manera que se agilice al máximo el algoritmo de búsqueda.

Imagen de la cara izquierda en escala de grises:

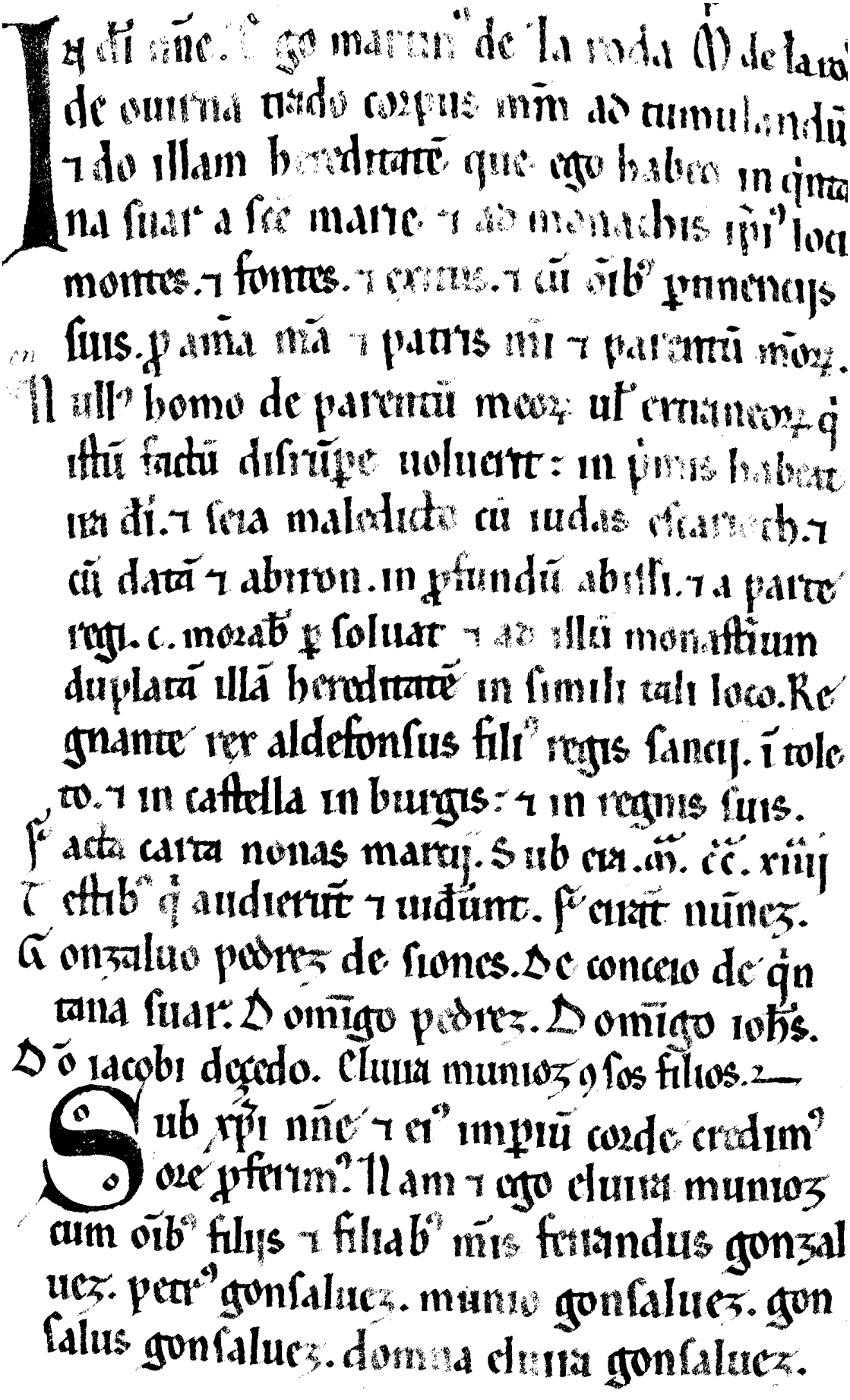


9.4 Segunda binarización

El último paso para obtener una imagen limpia y clara es realizar nuevamente la binarización que se empleó al principio eliminando así los píxeles grises. Este resultado también se ha obtenido utilizando dos programas de edición gráfica en un único paso, es decir, binarizando directamente la imagen de partida. Los programas han sido Paint y Nero PhotoSnap Viewer.



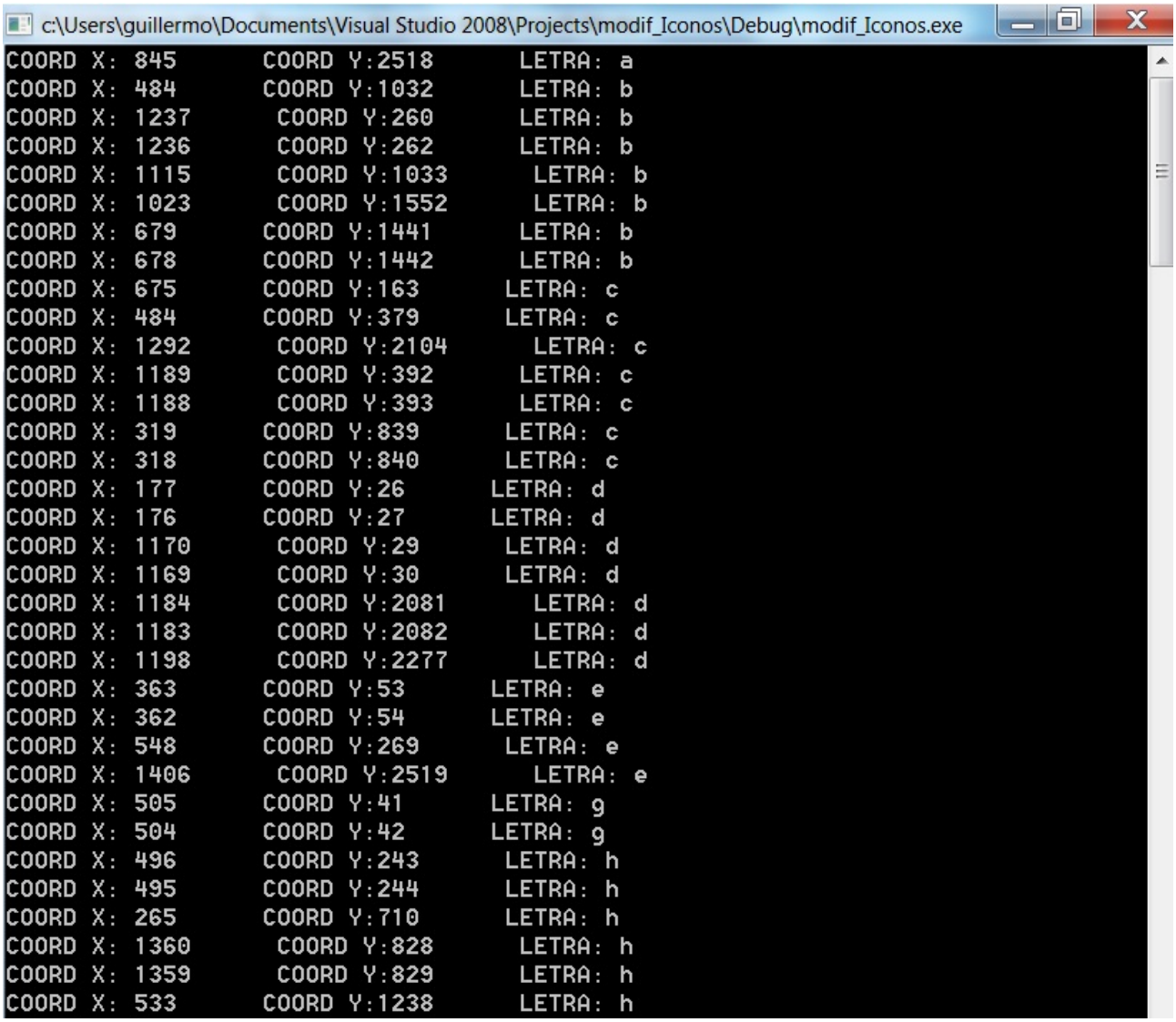
Ejemplo de binarización usando un programa de fotografía:



9.5 Grafías identificadas tras la búsqueda

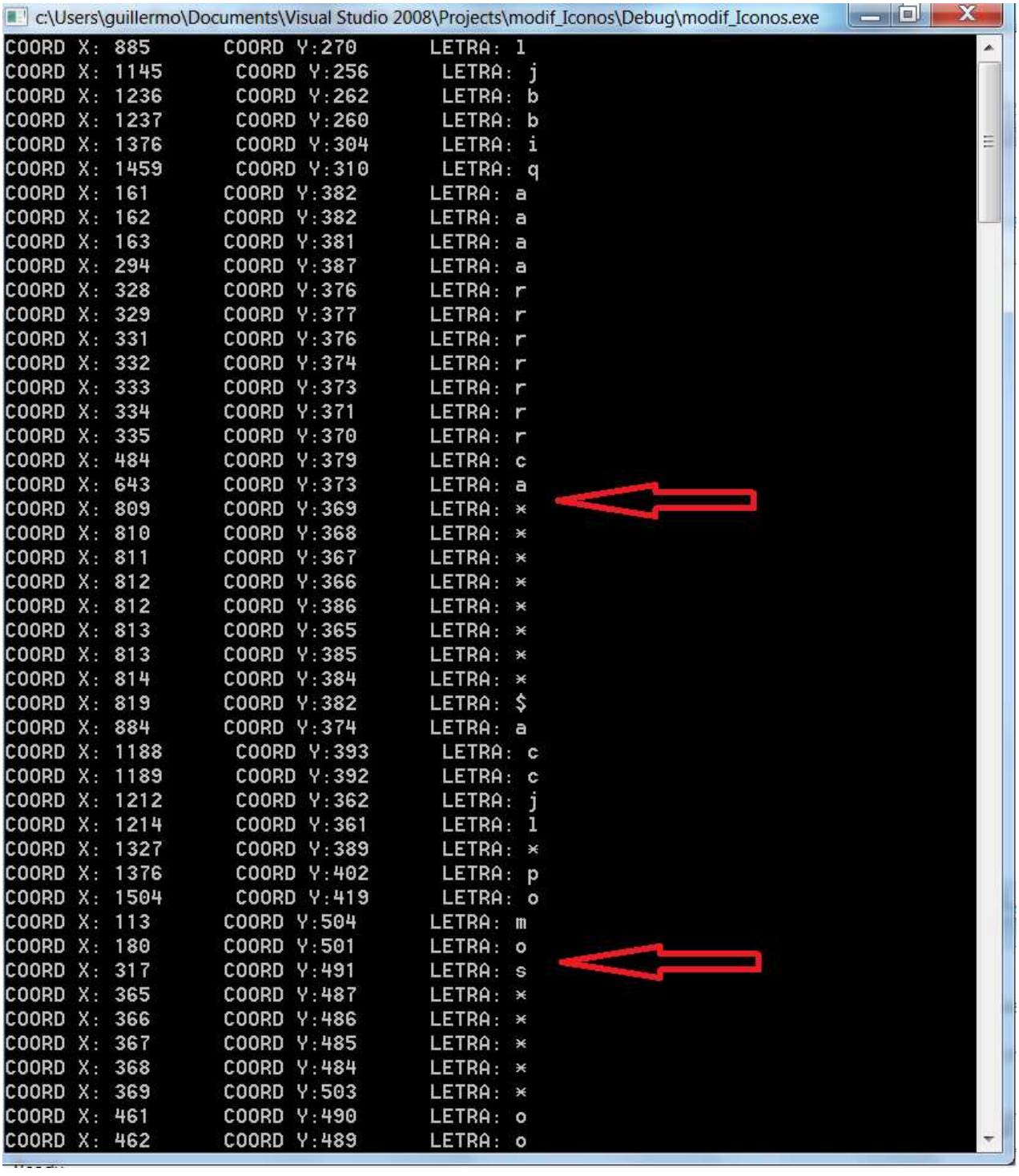
Tras la identificación de las grafías por plantillas, el programa muestra las coincidencias encontradas. Cabe resaltar que la asociación de cada letra con sus coordenadas espaciales es imprescindible para su posterior ordenación.

El umbral elegido ha sido de 0.16, es decir que el porcentaje de similitud entre la plantilla y la imagen ha de ser del 84%. Aunque este porcentaje pueda no parecer demasiado preciso, hay que señalar que al tratarse de un documento manuscrito, la variabilidad existente entre grafías, incluso de una misma letra, es muy elevada. Por tanto se hace necesario sacrificar precisión por porcentaje de grafías identificadas.



9.6 Ordenación de las grafías

A partir de los datos mostrados, puede observarse la casi inexistencia de una norma general para la ordenación de las grafías. Las letras están ordenadas de mayor a menor por el valor absoluto de sus abscisas y sus ordenadas, sin embargo, pueden observarse numerosas excepciones marcadas en rojo.

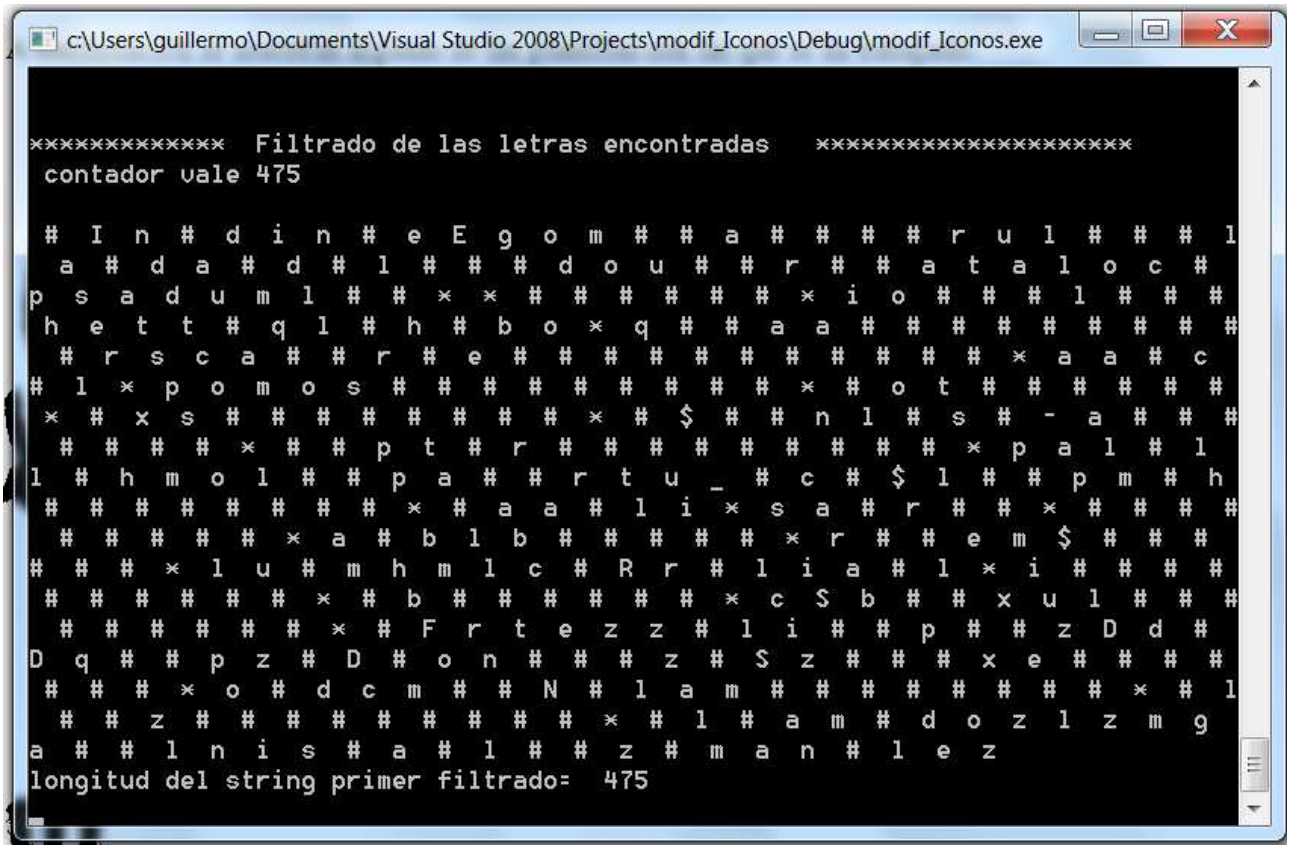


9.7 Filtrado de los datos

9.7.1 Primer Filtrado

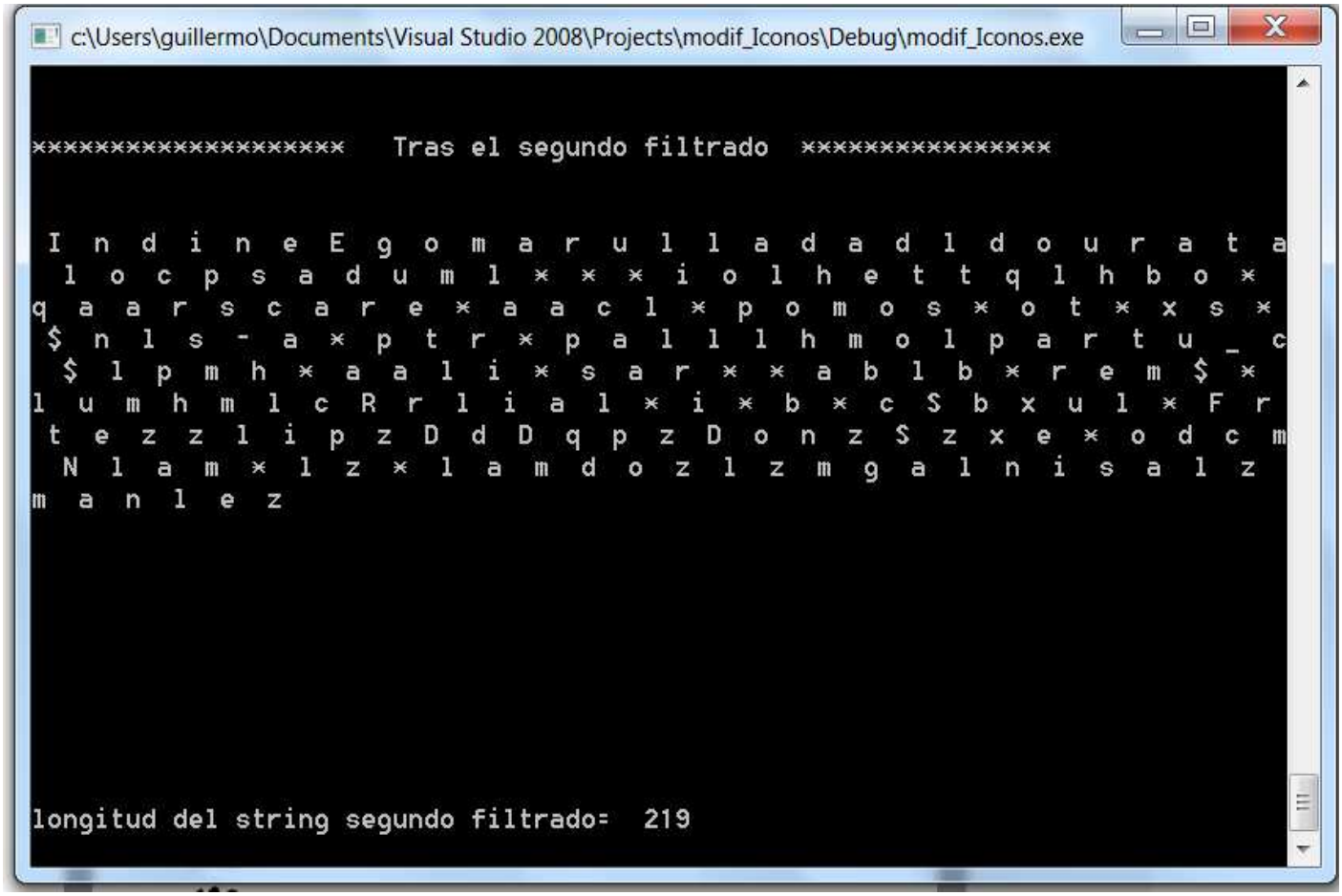
Tras esta operación se observa como varias letras han adoptado el carácter “#”, lo que corresponde a todas aquellas que están repetidas. Se decidió considerar como tales solo a aquellas que distasen menos de dos píxeles entre si y que tuvieran asignada la misma grafía. Se exceptuó de esta norma a las posibles dobles consonantes, como la “r” o la “l”. En estos casos, para su omisión, debían aparecer tres letras consecutivas con una distancia menor de dos píxeles entre ellas.

No es un método infalible, ya que en ocasiones aparecerán trascripciones de dobles consonantes que no figuraban en la imagen original, pero salvando este caso, con esté algoritmo se elimina la totalidad de letras repetidas.



9.7.2 Segundo filtrado

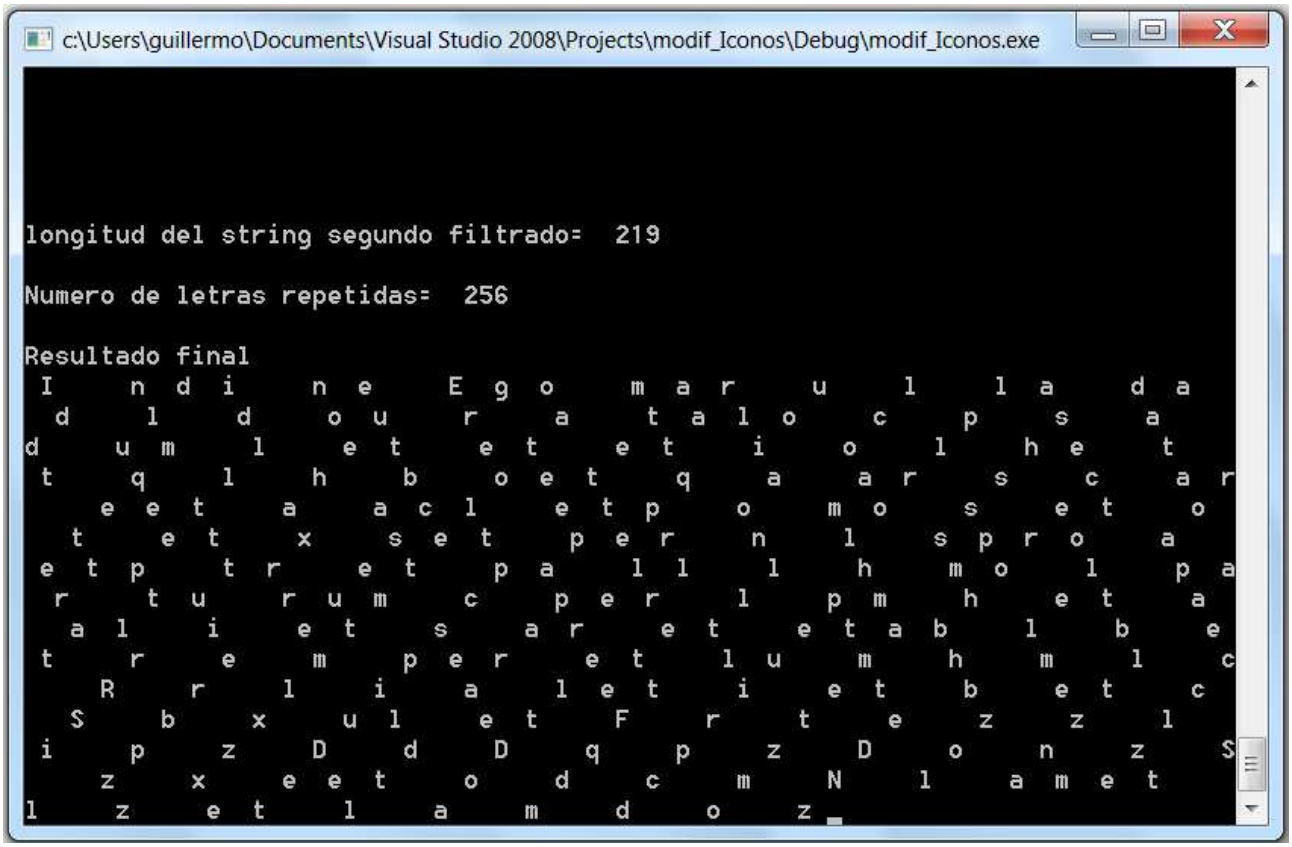
Esta operación tiene como objetivo localizar y eliminar los caracteres especiales “#”. Se decidió realizar la labor de filtrado en dos algoritmos diferentes por resultar más fácil.



9.8 Espaciado de los datos e Interpretación de los caracteres especiales

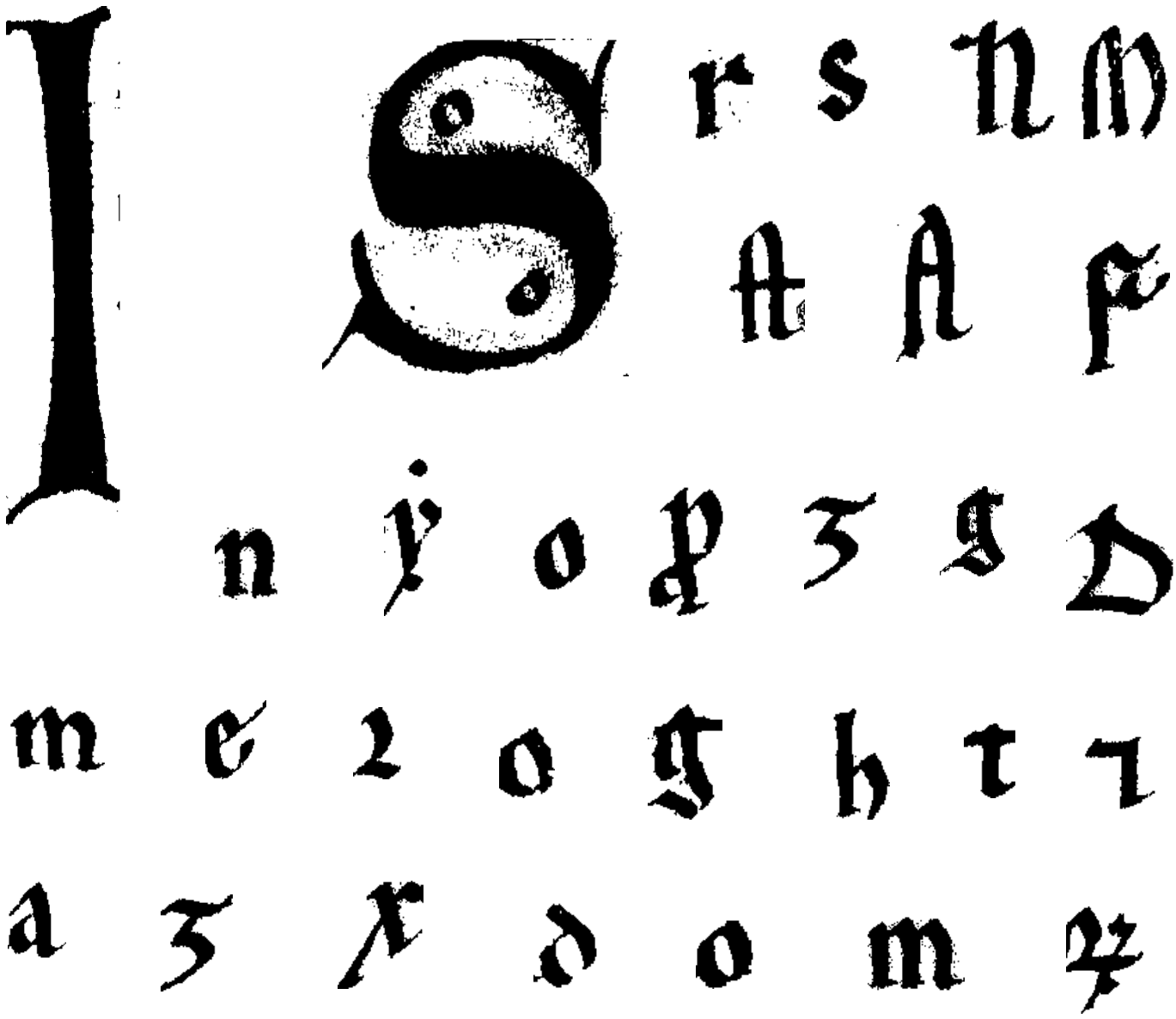
Mediante el algoritmo explicado en el apartado de métodos, el programa localiza los espacios en blanco existentes entre las grafías. Resulta bastante fiable pero al estar basado en un criterio de distancias entre letras, no es del todo preciso, ya que al tratarse de un texto manuscrito, las irregularidades tanto en el trazo como en el espaciado, son constantes. Por este motivo se ha decidido presentar el algoritmo alternativo, del cual se hizo mención en el apartado de métodos.

Como ha podido observarse en los datos expuestos, han ido apareciendo ciertos caracteres sin correspondencia ortográfica tales como el $ o el *. Recordemos que estos pertenecen a las contracciones y aféresis presentes en el texto de la imagen. Se ha realizado una interpretación de los mismos apareciendo en los resultados mostrados a continuación con su significado completo, sin abreviarse.



9.9 Ejemplo de plantillas utilizadas

A continuación se muestran algunas de las plantillas con las que se ha trabajado.



10. Conclusiones

En vista de los resultados obtenidos, se puede afirmar que:

1. Se trata de un programa todavía en desarrollo. El porcentaje de decodificación medio por línea es del 57.8%.

2. Los algoritmos de ordenación, filtrado, espaciado e interpretación de aféresis y contracciones funcionan de forma altamente satisfactoria.

3. Los algoritmos citados cometen un muy bajo porcentaje de errores.

4. Contribuyen a minimizar el tiempo total de decodificación de la imagen, puesto que son ejecutados a gran velocidad.

5. La calidad del algoritmo de búsqueda está directamente relacionada con el número de plantillas con el que se cuente. El porcentaje de grafías encontradas aumentará tanto en cuanto lo haga el número de plantillas.

6. Son necesarias un gran número de plantillas debido a las irregularidades caligráficas presentes en el texto.

7. A pesar de la gran tarea de iteración del programa, este se ejecuta en algo menos de tres minutos.

11. Trabajos futuros

Algunas de las posibles mejoras o ampliaciones que podrían llevarse a cabo en el programa son las siguientes.

Una base de datos

La creación de una base de datos simplificaría la tarea de carga y ampliación de plantillas. Podría realizarse con MySql.

Mejora del algoritmo de espaciado:

Sería necesario un algoritmo que identificase los espacios en blanco por similitud con una plantilla y no por distancias, ya que es mucho más fiable. A continuación se muestra un posible ejemplo del mismo con su explicación.

Este algoritmo es más complejo que el utilizado en el código, ya que debe localizar mediante plantillas que representen los espacios en blanco, únicamente las separaciones entre palabras y no los interlineales. Para ello, aunque utiliza las mismas funciones que el decodificador de grafías, a saber cvMatchTemplate y cvGet2D, es necesario que supere un considerable número de condicionantes, para que el algoritmo establezca que ha encontrado un espacio en blanco entre dos palabras. El siguiente procedimiento ha sido comprobar mediante un segundo umbral, "threshold_espacio", y la función cvGet2D, si el algoritmo localiza un espacio en blanco. De ser así, pasa a comprobarse a continuación si hay una letra antes o después del espacio en blanco localizado. Superada esta condición, se evalúa la existencia de espacios en blanco en las diagonales de la cuadrícula identificada como posible separación.
Detalle de las variables empleadas:

CvScalar s; // Localiza espacios en blanco
CvScalar antes_esp1x; // Evalúa si hay una letra antes del espacio.
CvScalar antes_esp2x; // Evalúa si hay una letra después del espacio.
CvScalar antes_esp1y; // Evalúa si hay una letra sobre el espacio.
CvScalar antes_esp2y; //Evalúa si hay una letra por debajo del espacio.
CvScalar antes_esp1xy; //Evalúa si hay una letra en el cuadrante inferior izquierdo del espacio.
CvScalar antes_esp2xy; //Evalúa si hay una letra en el cuadrante superior derecho del espacio.
CvScalar antes_esp3xy; //Evalúa si hay una letra en el cuadrante inferior derecho del espacio.
CvScalar antes_esp4xy; //Evalúa si hay una letra en el cuadrante superior izquierdo del espacio.

```
//Búsqueda de espacios en blanco

base_esp[0].img = cvLoadImage(espacio,0);

        w = base_img[0].img->width  - base_esp[0].img->width  + 1;
        h = base_img[0].img->height - base_esp[0].img->height + 1;


res = NULL;
res = cvCreateImage( cvSize( w, h ), IPL_DEPTH_32F, 1 );


cvMatchTemplate( base_img[0].img, base_esp[0].img, res,  CV_TM_SQDIFF_NORMED);


for( y = 0 ; y < h ; y++ ) {
  for( x = 0 ; x < w ; x++ ) {

   s = cvGet2D( res, y, x );
        if((y>1)&& (x>1)){
                anteriory=cvGet2D( res, y-1, x );
                anteriorx=cvGet2D( res, y, x-1 );}

        else {anteriory=cvGet2D( res, y, x );
        anteriorx=cvGet2D( res, y, x );
        }
        if((x>7)&& (x <  res->width-7)){
                antes_esp1x=cvGet2D( res, y, x-6 );
                antes_esp2x=cvGet2D( res, y, x+6 );}
        else{
                antes_esp1x.val[0]= 0.08;
                antes_esp2x.val[0]= 0.08;}
        if ((y >5) && (y <res->height-5)){
                antes_esp1y=cvGet2D( res, y-4, x );
                antes_esp2y=cvGet2D( res, y-4, x );}
        else {
                antes_esp1y.val[0]= 0.8;
                antes_esp2y.val[0]= 0.;}
        if((x>8)&& (x <  res->width-8) && (y >8) && (y <res->height-8)){
                antes_esp1xy=cvGet2D( res, y-3, x-3 );
                antes_esp2xy=cvGet2D( res, y+3, x+3 );
                antes_esp3xy=cvGet2D( res, y-3, x+3 );
                antes_esp4xy=cvGet2D( res, y+3, x-3 );
        }
        else{
                antes_esp1xy.val[0]= 0.8;
                antes_esp2xy.val[0]= 0.8;
                antes_esp3xy.val[0]= 0.8;
                antes_esp4xy.val[0]= 0.8;}}
```

```
                if( s.val[0] <= threshold_espacio ) {//localizado un espacio en blanco

        if((antes_esp1x.val[0]>threshold_espacio)||(antes_esp2x.val[0]>threshold_espacio)){
                        //es decir, que no supera el umbral xq hay una letra antes o despues


        if((antes_esp1y.val[0]<=threshold_espacio)||(antes_esp2y.val[0]<=threshold_espacio)){
                        //es decir, si supera el umbral, es porque hay un espacio arriba o  abajo
/*if(((antes_esp1xy.val[0]<=threshold_espacio)&&(antes_esp2xy.val[0]<=threshold_espacio))||
((antes_esp3xy.val[0]<=threshold_espacio)&&(antes_esp4xy.val[0]<=threshold_espacio))){
                        //al menos dos de los cuatro parámetros debe ser un espacio en blanco*/


if(( anteriory.val[0] <= threshold_espacio )|| (anteriorx.val[0]<=threshold_espacio)) {
                                        //significa que ya he encontrado otro espacio en blanco
        x++;}




                        else{
                                printf("Detecto espacio en blanco y recuadro\n");
                                cvRectangle( base_img[0].img,
                                cvPoint( x, y),cvPoint( x + base_esp[0].img->width, y +
                                        base_esp[0].img->height ),cvScalar( 0,0,0,0),2 , 0, 0 );

                                texto[contador+ incremento].posx = x;
                               texto[contador+incremento].posy = y;
                                texto[contador+incremento].l = '-';
                                incremento=incremento+1;
                                                }
                                                }
                                }
                        }
                }
}
contador=contador+incremento;
```

Como puede observarse, se ha empleado otro contador, “incremento”, para posicionar los espacios en blanco situados a continuación de las grafías en el array de estructuras “texto”.

Mayor número de plantillas:

Como ya se ha expuesto, debido a las irregularidades del texto, se hace necesario un elevado número de plantillas para su correcta decodificación.

Programa de segmentación de grafías:

Una mejora considerable en el programa sería adaptarlo para que fuera capaz de segmentar de forma fiable, las letras de una palabra. De esta manera podrían decodificarse mayor número de textos, al tiempo que sería posible implementar distintos métodos de búsqueda. Por ejemplo que el programa pudiera completar algunas de las palabras decodificadas, basándose en un diccionario. Para ello debe ser capaz de segmentar el fragmento de palabra decodificado y si supera un porcentaje mínimo de grafías decodificadas, se compara con palabras similares del diccionario.

Binarización de la imagen por áreas:

Debido a que el material con que está elaborado el códice es piel animal, las tonalidades a lo largo de la hoja varían. Por tanto el valor umbral para su correcta binarización no es único, sino que depende de la región de la hoja en la que nos encontremos. Por este motivo creemos, que otra posible mejora del programa sería implementar un código que calculase el valor óptimo de umbralización según el área analizada.

12. Bibliografía

AGAM, Gady. *Introduction to programing with OpenCv.* Department of computer Science Illinois Institute of Technology. 2006

CARRETERO, J., F. García, J. Fernández, A. Calderón. *El lenguaje de programación C: diseño e implementación de programas*. Prentice Hall, 2001.

Imágenes JPEG, PNG y BMP. Wikipedia enciclopedia libre.

Monasterio de Santa María de Rioseco. Wikipedia enciclopedia libre

Open Source Computer Vision Library: Reference Manual. Intel Corporation. 2006

The Medieval Miniaturæ Compendium. DGSCA/UNAM. 1997-1999. URL

RIESCO TERRERO, Ángel. *Introducción a la Paleografía y a la Diplomática General*. Madrid: Síntesis, 2004.